マシン語の基礎からインベーダーまで

たのしいエレクトロニクス・ホビー

PC-8001/8801 マシン語入門

N-BASICモード

マシン語の基礎からゲームの製作まで

10 ホビーライフシリーズ

バトルファイアー

アスロック

プラネットバルカン

スーパーバルーン

ファイアー・インフェルノ

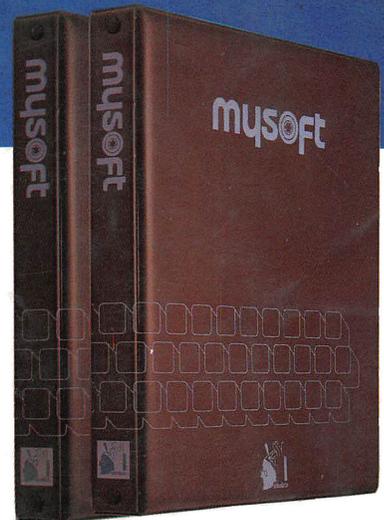
脱獄ゲーム

JOHO-CENTER

ホビーライフ 10

PC-8001/8801 マシン語入門

電波新聞社

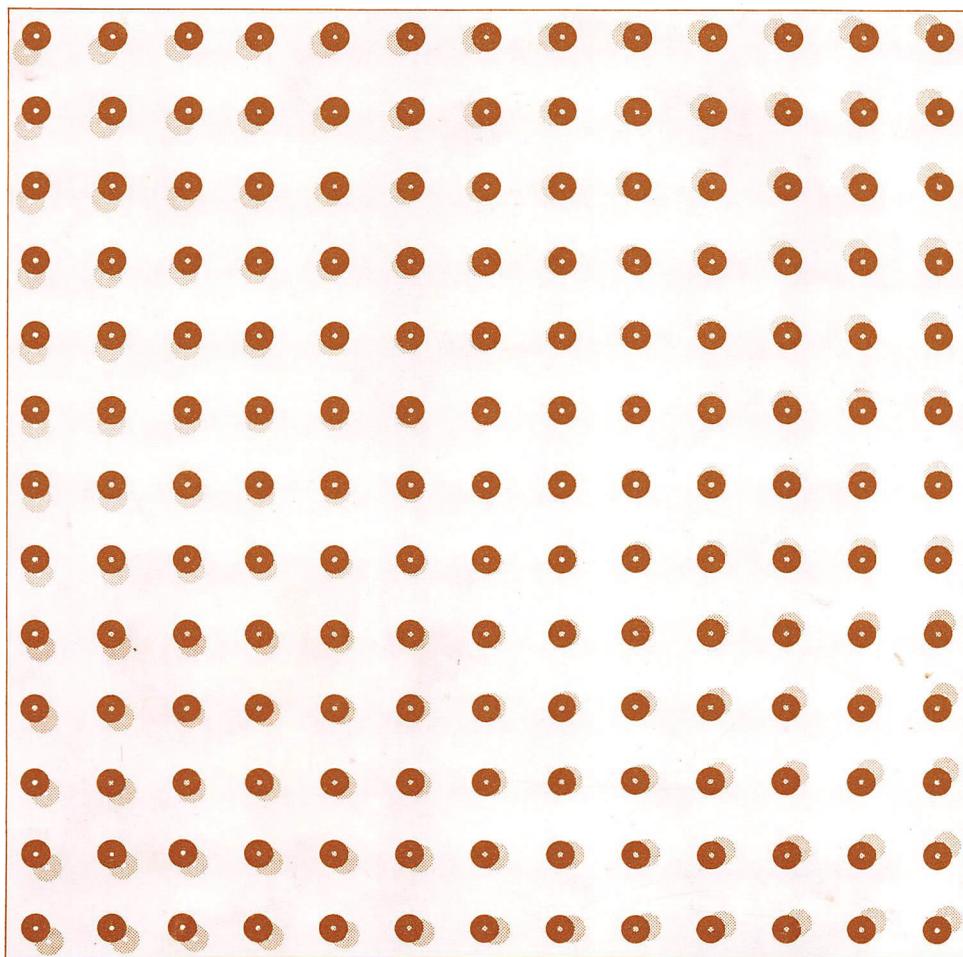
電波新聞社 〒141 東京都品川区東五反田1−11−15 電話(03)445−6111（大代） 定価1,300円 雑誌08370-7

マシン語の基礎からインベーダーまで

たのしいエレクトロニクス・ホビー

PC-8001 8801 マシン語入門

N-BASICモード

マシン語の基礎からゲームの製作まで

C&C
コンピュータ・コミュニケーション
キャリアに
パソコンフィールド、
一気に拡大。
NEC N5200
0
1

差、NECの16ビット。

4機種のラインアップで幅広い用途に応えるNECのパソコンファミリー。
愉しい家庭用から、ビジネスのフットワーク
オフィスの強力なシステムへと、活躍の場がグングン拡がっています。
おなじみの8ビットと、技術の蓄積をもつ16ビットの中か
あなたにぴったりの1台をお選びください。

先進の16ビット

N5200 モデル05

●高性能16ビットCPU(μPD8086)を採用●主記憶装置が256Kバイトと大容量●BASIC、COBOL●高級カラーグラフィック●OAソフトウェアパッケージ●高度な日本語処理●オペレーティングシステム<PTOS>の他、CPM-86,* MS-DOS*の利用可●本格的通信機能●1Mバイト薄型フロッピィディスク●インテリジェントターミナル機能●**システム標準価格:698,000円**(ディスプレイ、キーボード、フロッピィディスク1台)

※CPM-86はデジタルリサーチ社、MS-DOSはマイクロソフト社の登録商標です。

洗練の8ビット

PC-8800シリーズ

●強力なN88-BASICを搭載、新たな周辺機器も加えてアプリケーションの可能性をさらに拡大●PC-8001の各種周辺機器ソフトウェアをそのまま利用可能●漢字ROM(オプション)の使用により、日本語の文書作成が容易●標準実装184Kバイトの余裕あるメモリ●最高640×400ドットの高解像度●充実のグラフィック機能●**本体標準価格:228,000円**

PC-8000シリーズ

●国内実績No.1の人気機種●強力な周辺機器があらたに加わり、シリーズの内容がさらに充実●8色のカラー表示と8段階の濃淡による、良質の見やすい画面を実現●豊富なアプリケーションソフトウェアがそろった実績のある応用力●各種インタフェースを内蔵して拡張自在●ターミナルとして使用可能●**本体標準価格:168,000円**

PC-6000シリーズ

パピコン

●2個のCPUを使い、機能充実●プログラムがカートリッジ化され、ワンタッチの入力で多機能を発揮●家庭用テレビ・ビデオモニターテレビにそのまま接続可能●シンセサイザー機能で、三重和音までの自動演奏●従来のコンピュータの文字・記号に加え、ひらがなの使用が可能●**本体標準価格:89,800円**

Bit-INN TOKYOシステムセンター
〒101 東京都千代田区外神田1-15-16
ラジオ会館7F ☎03(255)4006、4575~6

Bit-INN OSAKAシステムセンター
〒542 大阪市南区難波千日前13-11
マスザキヤビル4・5・6F ☎06(647)2747~8

Bit-INN NAGOYAシステムセンター
〒460 名古屋市中区大須4-11-5
杏林殖産ビル2F ☎052(263)0971

Bit-INN YOKOHAMAシステムセンター
〒220 横浜市西区北幸1-8-4
横浜西口第2ミナトビル7F ☎045(314)7707~9

NEC日本電気グループ

日本電気株式会社
パーソナルコンピュータ販売推進本部 〒108 東京都港区芝5丁目33-7(徳栄ビル) ☎(03)453-5511(大代)
端末装置事業部 販売促進部 〒108 東京都港区芝4丁目14-2(第二田町ビル) ☎(03)454-9111(大代)
●お問い合わせは、NEC及び最寄りのBit-INN、NECマイコンショップ、NEC商品販売(株)へ。

新日本電気株式会社
パーソナルコンピュータ・ディスプレイ事業部販売部 〒213 川崎市高津区久本210番地
☎(044)833-5201(大代)

触れることからはじまる

ゆとりのパソコンスペース

広びろとしたスペースで、ゆったりとパソコンを確かめてみませんか。実用を求める大人のためのパソコンショップです。

ソフトを選んで効率アップ!

Helps
実用標準ソフト

給与計算システム ¥80,000
財務会計システム ¥80,000

※随時デモンストレーションを行っていますので、
お申し込みください。

ビジネスのエキスパート
PC-8800シリーズ

実績の名機
PC-8000シリーズ

楽しめるパソコン
PC-6000シリーズ

目的に合わせたセミナー6月より開設!

NECマイコンショップ

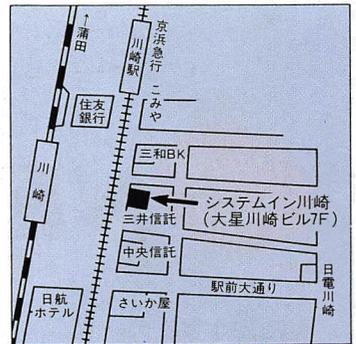
JMCシステムイン川崎

川崎市川崎区駅前本町5の2 大星川崎ビル

☎(044)211-2234

AM10:00〜PM6:30(日・祭日休み)

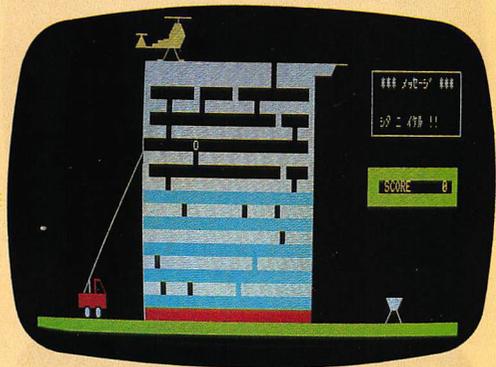
JMC 日本マイクロコンピュータ株式会社

本社 〒102 東京都千代田区麹町4-5-21睦ビル☎(03)230-0041(代)

Dempa

PC-8801 PC-8000

大紹介

マイコンソフト

ファイアー・インフェルノ

ビルが大火事さあたいへん。煙でだんだん通路が見えなくなる。7Fまで行けばハシゴ車に、屋上まで行けばヘリコプターに間にあうが…。煙が充満するともうハシゴ車もヘリコプターも使えない。今度は下から火が……。

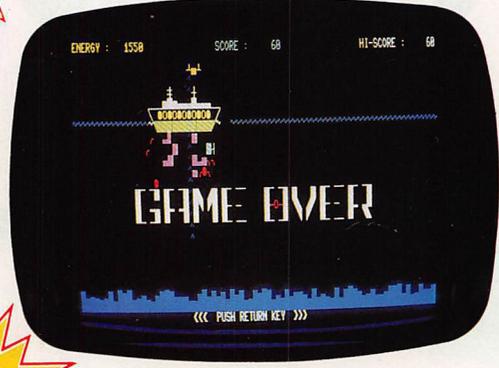
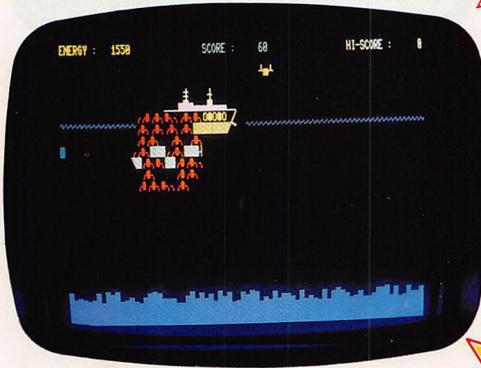
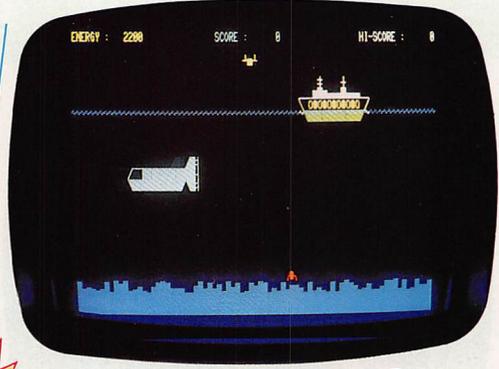
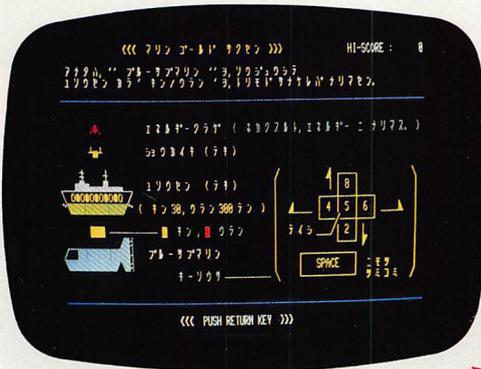
詳しくは当社ソフトテープ案内のページをご覧下さい。

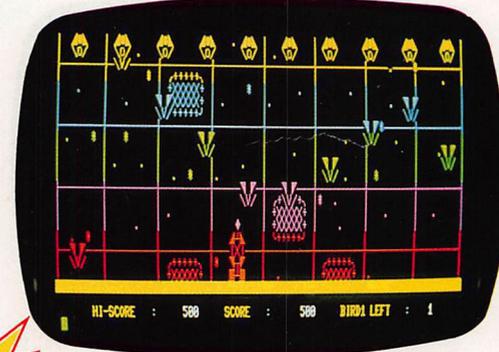
バトルファイアー

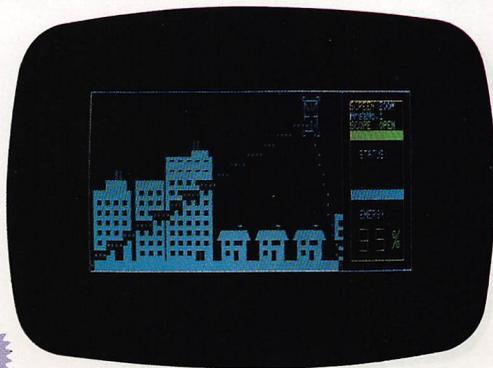
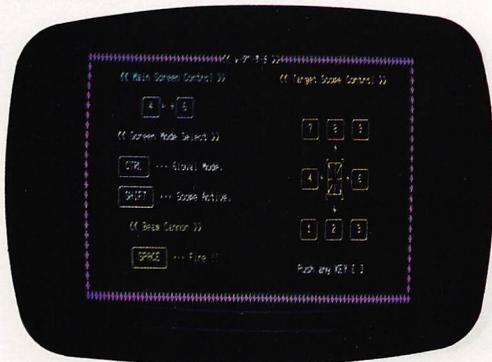
せまりくる敵大船団。ターゲットスコープオープン！自動追尾装置セットオン！目標をまちがえるな。サブロックの数には限りがある。敵船団を壊滅させろ!!

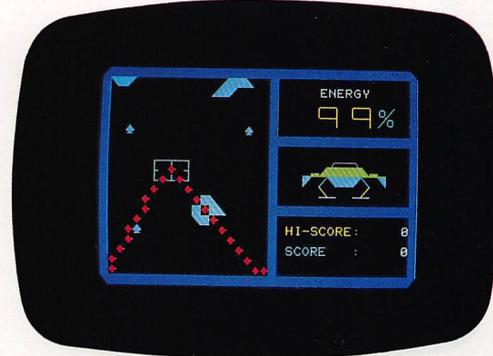
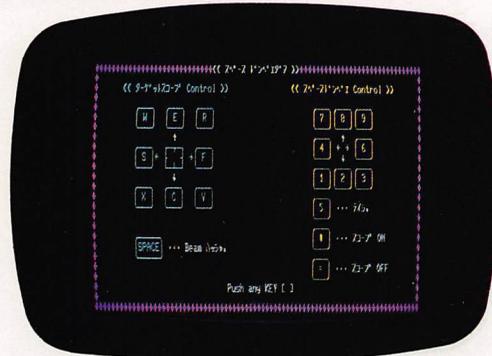
PC-8801・8001

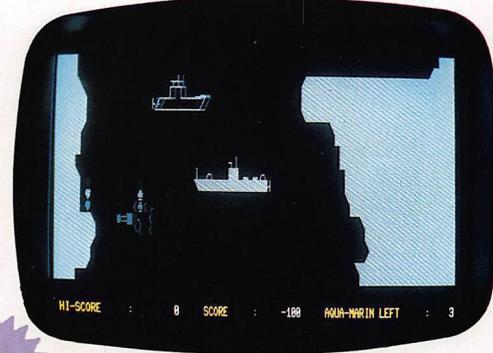
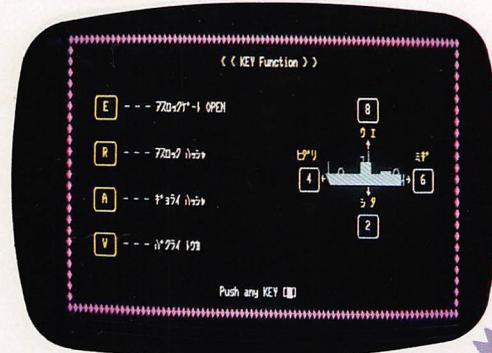
マリンゴールド作戦

某組織の手に依り、多量の金塊、ウランが盗み出された。彼等は高性能ソナーを搭載した哨戒機を護衛に付け、輸送船で逃亡。更に海中には恐るべきエネルギー水母が……。高速水中母艦ブルーサブマリン号に乗り込んだ君は、無事奪還することができるか！

フューチャー

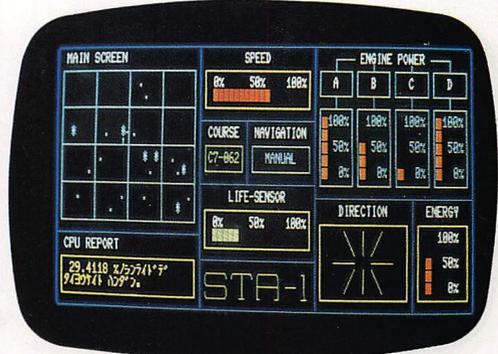
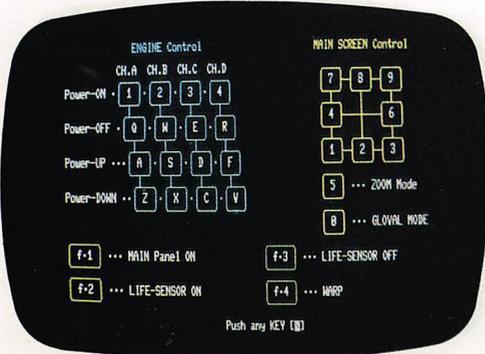
宇宙連合の命を受けて、君はバード号に乗り込み、敵ガメロンを退治しなければならない。突如出現するブラックホール、吸い込まれたら二度と生きては帰れない。君はガメロンを退治できるか。

レーダーサーチ

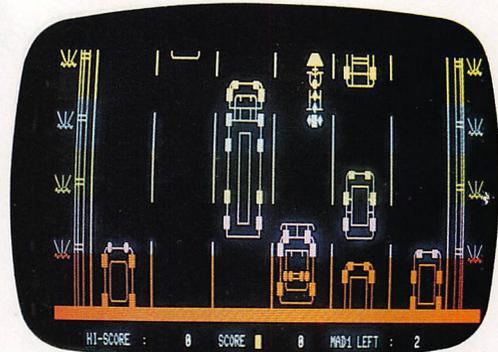
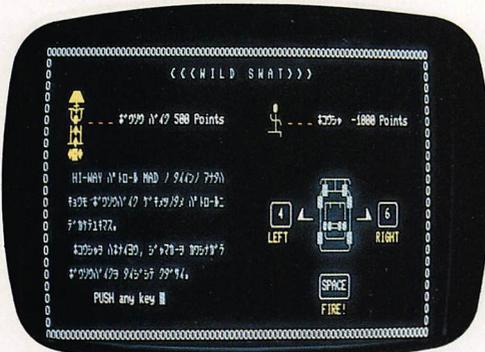
謎の宇宙人の手により地球上の主要都市が全て破壊されてしまった。最後のとりで、オアシス島にたてこもり、わずかに残されたレーダーを使って敵の侵略を防げるか！

スペースどんべえだあPART 1

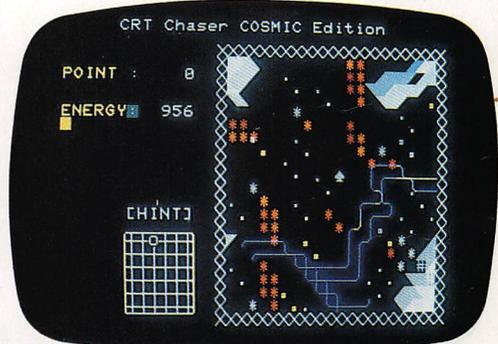
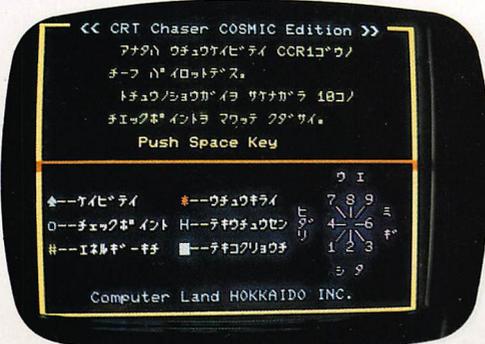
君は監視船どんべえ1号の船長。今日もアステロイドベルトのパトロールに出たが、エイリアンはずるがしこくて惑星の陰にかくれている。さあ、エイリアンを探し出せ！

アスロック

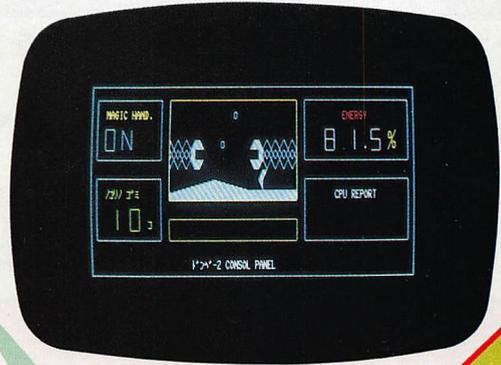
海上都市が立ち並ぶ超近代惑星・地球。ところがある日突然謎の海底人の手によって各国の都市が破壊されてしまった。立ち上がれアクアマリン号！

PC-8801・8001

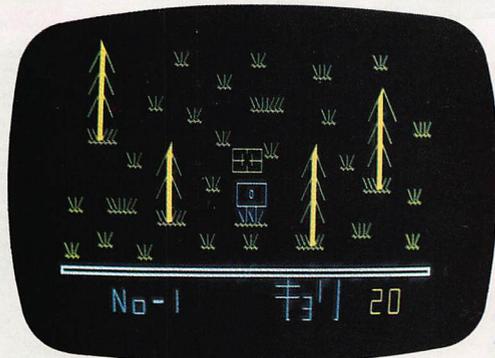
シリウスF

次元断層へ落ち込んでしまった宇宙船STA-1号。操縦席の君はコントローラを駆使し地球へ帰還できるか！

ワイルドスワット

町の中を我が物顔で走る暴走族。装甲パトカーMAD-Xに乗り込んだ君は暴走族絶滅に立ちあがった！

CRTチェイサー①

君は任務は宇宙空間の警備。機雷やミサイルに注意して無事10か所のチェックポイントを通過できるか！

マリンどんべえだあPART1

君は海洋廃品回収船どんべえII世号のチーフパイロット。廃品識別装置なしでゴミだけをかき集めよう！

エアーライフル

標的は3つ、距離はそれぞれ10m、20m、30mにセットされている。11発弾丸で君は何点かせげるか！

スカイどんべえだあPART1

大気汚染調査隊スカイどんべえは、今日も調査に出かけます。アッ！危い。その雲はオキシダントのかたまりだ!!

SYSTEM INN **KODENSHA**

NECと高電社がガッチリ握手、NECシステムイン高電社の誕生です。

NECシステムイン高電社の誕生で、ベストセラー機PCシリーズへのサポート体制がいちだんと強力になりました。フロアにはPCシリーズ全機種をはじめ、さらにグレードを高める周辺機器も豊富にラインナップ。また、オリジナリティー豊かなソフト開発に取りくみ、PCシリーズの可能性を大きく拡げます。パソコンの最新情報があふれるNECシステムイン高電社へ、ぜひ一度お越しください。

選べる、3機種、3機能 国内実績No.1

NECのパソコン

PC-8800シリーズ ¥228,000(本体価格)

PC-8800(N88ベーシック)

- PARAM-K1 ——— ¥49,000
- PARAM-3 ——— ¥39,000
- graph-88 ——— ¥19,000
- 日本語ワープロ マイレター ——— ¥49,000
- 英文ワードプロセッサー ——— ¥33,000

PC-8000シリーズ ¥168,000(本体価格)

PC-8000(Nベーシック)

- PARAM-1 ——— ¥39,000
- PARAM-2 ——— ¥39,000
- graph-7 ——— ¥19,000
- 見積実行予算システム ——— ¥90,000
- 英文ワードプロセッサー ——— ¥33,000

PC-6000シリーズ ¥89,800(本体価格)

テープ:40種類 各¥2,800

●ハコイリムスメ●トレック●ハノイ●ミグ15●フリップ●ホシトリゲーム●メイロ●サイモン●ゴルフ●チェッカー■歴史年表■漢字他

パピコン

NECマイコンショップ

南大阪にいまシステムイン高電社。

『やっぱり日本人は、漢字ですね。』

パソコン用簡易言語・パラムに、いま漢字仕様のパラムK新登場!

大好評の簡易言語・パラムシリーズに、いま漢字の使える パラム K が登場しました。 誰にでもプログラムづくりができる手軽さに、漢字の見やすさ、読みやすさが加わっていっそうパソコンを使いやすくしました。また、漢字ソフトシリーズには待望の漢字ワードプロセッサーソフト「マイレター」も新登場。パソコンをいちだんと機能アップさせます。

PARAM/K1

使用機種
PC-8800・FM8

ディスクベース **¥49,000**

1. 項目(データ名)の数と長さ、画面、プリンター出力が自由設定できます。
2. 並べかえ、追加、修正、削除は簡単。
3. 1件(1レコード)64文字から128文字まで。
4. 複合条件(AND. OR. NOT)で検索します。
5. 1行は漢字仕様で53文字。
6. 複合条件(例えば東京都、男性、25才以上、未婚)で必要なデータを検索して表示印刷します。

簡易文書作成機能もついて、即実務に活用出来ます。

「文書づくり」の煩しさともさようなら!!

漢字ワープロ〈マイレター〉待望の新登場

マイレター

使用機種
PC-8800

ディスクベース **¥49,000**

―日本語ワードプロセッサー"マイレター"の概要―

〈カナ漢字変換方式〉

1. 使用文字種 漢字JIS第1水準2965文字、その他500文字
2. ディスプレイ表字文字数 40字×20行(10行)
3. 単語辞典10000語(オプション)。登録可能・単語は8文字(熟語)まで。
4. 訂正・挿入・削除は簡単。
5. ブロック移動・タブ、可能。
6. アンダーライン、センターリング、可能。
7. 枠あけ、差込み、罫線引きが出来ます。
8. 登録、呼び出し、文書保存が出来ます。
9. 改行ピッチ・コントロール、字間ピッチ・コントロールが出来ます。
10. 行挿入・消去が出来ます。
11. 禁則処理有

GRAPH/7

〔自動グラフ作成プログラム〕

使用機種 PC-8000
ディスクベース
¥19,000

英文ワードプロセッサー〔ワード9000〕 ●フログラム価格 ¥ 33,000

ハングル漢字ワードプロセッサー〔ハングル4300〕 ●フログラム価格 ¥ 155,000

見積・実行予算システム〔エスコ2000〕 ●フログラム ¥ 90,000

高電社ではこのほかに各種実用フログラムを豊富にとりそろえております。

系統的なカリキュラムと充実した教育内容を誇る
高電社パソコン学院

科目	期間	受講料
ベーシック入門科	毎月初旬開講 週1回・8週修了 (昼1:30～4:00 夜6:20～8:50)	¥25,000
ディスク応用科		¥45,000
プログラミング演習科		¥30,000
アセンブラ科		¥35,000
実務入門科		¥35,000

●本社教室(国鉄天王寺より送迎バス)、大阪駅前第4ビル教室があります。
■実務入門科は週1回・4週修了コースもございます。■入学金、各科1律¥5,000

●DMのあて名書き代行業務うけたまわります。

●マイコン・技術者募集中! 岩城迄。

※なお詳細は電話でお問合わせ下さい。

システム販売からソフト開発、オペレーション教育まで 高電社

NEC:システムイン高電社 〒546 大阪市東住吉区杭全7-10-15―― TEL(06)719-1131(代)

高電社パソコン学院 〒530 大阪市北区梅田1-11-4 大阪駅前第4ビル6F TEL(06)341-3371(代)

はじめに

ヤング・パワー――この言葉は，まさにマイコンの世界のためにあるが如く，若き少年達の凄まじきパワーには圧倒されるものがあります。

マイコン・ショップで見せる彼等の華麗な指さばき。

マイコン雑誌は彼等のハイテクニックの品評会であり，その

吸収力の速さ

情報量の豊かさ

アイデアの柔軟性

は，大人達の比ではありません。

あなたは，自分のマシンを使いこなしていますか？

自由に付属のマニュアルを理解できますか？

もし何かひっかかるものがあるとすれば，それはもしかしたら

マシン語音痴

によるものかもしれません。

本書は，PC―8001，PC―8801を対象としたZ―80マシン語の解説書です。しかし，

本書を読んでも

マシン語ができるようにはならない！

でしょう。驚きましたか？　驚いた人は，次を読んでみてください。

BASICを理解できる人は，マイコン所有者の10分の1と言われています。さらにマシン語まで理解できる人は，その10分の1と言われています。すなわちマシン語は，解説書をチョロッと読んだくらいで理解できるようなそんな生易しいものではないのです。

ところで

他書を読んでもマシン語が理解できなかった人

本書を読んでみてください。たぶんあなたは，

マシン語を理解できるようになる！

でしょう。

今度は逆説的みたいなことを書きました。しかし今やあなたは，

マシン語の難しさ

を知っています。決して

マシン語をあなどる

ことはしないでしょう。そんなあなたが**マシン語修得への熱意**をかけて，本書を片手に努力してくださるなら，本書はあなたに

マシン語への近道

を提供することになるでしょう。なぜなら本書は，マシン語初学者がかかえる

マシン語固有の問題点からの解放

への願いをこめて，非力な私が全力を傾けてみたものだからです。

本書は，非常にぜいたくな構成をとっています。

① 物理的な制限を設けていません。
　わかりにくい部分，理解しにくい部分にはページ数の糸目をつけず，また図もふんだんに入れて解説を試みました。

② 対象にも制限を設けていません。まったくの初心者でも理解可能です。

③ 最終目標もぜいたくです。それは，

オール・マシン語カラー・グラフィック版で，スペース・インベーダーなみのゲームの制作

においています。マシン語のプログラムでもこれだけの規模になると，相当に長く，大きいものになります。これならマニアの方にも満足していただけるのではないでしょうか？

ところで以上の構想の下に原稿を進めると，とても1冊では収まりきれません。したがって本書は続編と

シリーズ(全3冊)

を成すものであり，シリーズの入口に当るものです。本シリーズがあなたのマシン語学習への手助けとなり，あなたの**夢の実現**に少しでもお役に立てれば，幸いに思います。また，本書を読まれるすべての人が，途中くじけることなく進めるようお祈りしています。

最後に本書の出版にあたり，電波新聞社出版部の皆様，また FORESIGHT の仲間に大変お世話になりました。ここに慎んでお礼申し上げます。

1982年５月16日

FORESIGHT企画部

塚越一雄

もくじ

モニタ・オペレーションの実践

第1章　マシン語のコマンド・レベルを求めて

第2章　TM・Sコマンドの中身を探る

第3章　マシン語のプログラムGO！

第4章　カセット・オペレーションとメモリ・エラー

第2ブロック

マシン語プログラミングの実践1

第1章　マシン語のプログラムに初挑戦

第2章 ハンド・アセンブルの実践

第3章 レジスタを求めて

第3ブロック マシン語プログラミングの実践2

第1章 画面表示への2大要素

第2章 画面表示の実現

第3章 そして全てが消えた

もくじ

第4章　マシン語ユーティリティの開発

第4ブロック　マシン語の散歩道

第1章　さらに詳しく知りたい人のために

第2章　インベーダーの1列編隊

付録

第1ブロック

モニタ・オペレーションの実践

```
NEC PC-8001 BASIC Ver 1.2
Copyright 1979 (C) by Microsoft

Ok
mon
*DFF39,FF3C
FF39 20 20 20 20
*DD000,D02E
D000 00 01 02 03 04 05 0C 2C
D008 20 FB 7E B9 20 12 0C 2C
D010 20 FB 0C 20 F0 25 ED 57
D018 BC 38 E8 28 E8 C3 00 00
D020 32 3C FF 79 32 3B FF 22
D028 39 FF 3E 21 D3 40 76
*
```

も・に・た——って何？

第1章 マシン語のコマンド・レベルを求めて

〈はじめに〉

いよいよ本章からマシン語の話が始まります。**「マシン語」**ってどんな形をしているのでしょう？

我々は，まず**マシン語を観察する**ことから始めます――**注意深く，注意深く観察**してみてください。マシン語を観察するだけでもいろいろなことがわかってきます。

次いでマシン語を入力することを試みます。すると，

Syntax error

が出て思わぬ障害に遭遇することになります。そしてその障害を乗り込えるため，我々は**マシンの制御をモニタに移す**ことを知るでしょう。

1. マシン語のプログラムを観察する

マシン語への旅――まず手はじめに

マシン語そのもの

を眺めてみましょう。これからマシン語をいろいろな形で示します。本ブロック終了時には，どんな形のマシン語のプログラムでも，あなたは

自由にあやつれる

ようになっていることでしょう。

まず**第1-1図**をみてください。

D000	21	00	FF	0E	00	71	0C	2C
D008	20	FB	7E	B9	20	12	0C	2C
D010	20	F8	0C	20	F0	25	ED	57
D018	BC	38	EA	28	E8	C3	00	00
D020	32	3C	FF	79	32	3B	FF	22
D028	39	FF	3E	21	D3	40	76	00

《第1－1図》マシン語のプログラムの形――その1

これが，マシン語のプログラムです。良く観察してみてください。**英数字**が並んでいますね。英字は，どんな種類が使われていますか？

次に**第1-2図**をみてください。これもマシン語のプログラムです。そして，**第1-3図**もみてください。もちろん，これもマシン語のプログラムです。

三つのマシン語のプログラムを見てみました。どれ

D000	21 00 FF 0E 00 71 0C 2C 20 FB 7E B9 20 12 0C 2C
D010	20 F8 0C 20 F0 25 ED 57 BC 38 EA 28 E8 C3 00 00
D020	32 3C FF 79 32 3B FF 22 39 FF 3E 21 D3 40 76 00

《第1－2図》マシン語のプログラムの形――その2

番地	マシン語	番地	マシン語
D000	21	D018	BC
D001	00	D019	38
D002	FF	D01A	EA
D003	0E	D01B	28
D004	00	D01C	E8
D005	71	D01D	C3
D006	0C	D01E	00
D007	2C	D01F	00
D008	20	D020	32
D009	FB	D021	3C
D00A	7E	D022	FF
D00B	B9	D023	79
D00C	20	D024	32
D00D	12	D025	3B
D00E	0C	D026	FF
D00F	2C	D027	22
D010	20	D028	39
D011	F8	D029	FF
D012	0C	D02A	3E
D013	20	D02B	21
D014	F0	D02C	D3
D015	25	D02D	40
D016	ED	D02E	76
D017	57	D02F	00

《第1－3図》マシン語のプログラムの形――その3

を見ても英数字の羅列でしたね。そして，三つとも異なる形をしていました。実は，

これら三つのプログラムは

同じ内容のプログラム

です。

もう少し観察を繰り返してみましょう。

三つのプログラムのどこを見ても英数字が並んでいますが，

4桁の英数字

2桁の英数字

の2本立てになっています。そして，2桁の英数字を順に見ていくと，

21　00　FF　……

と三つとも同じ並べ方になっているのがわかります。また，4桁の英数字の先頭は三つとも

D000

になっていますね。これが

三つとも同じプログラム

であることの証明です。

2. 英数字の正体

前節で見てきた**4桁の英数字**を

番　地

といい，2桁の英数字を狭義の（しかし，本来の意味での）

マシン語

と言います。ただし，ここでこれらの言葉を覚える必要はまったくありません。

次に，英数字の部分をもう少し解析してみます。もう一度，三つのプログラムを良く観察してみてください。とくに英数字の部分に注意して。

良く見ると英字は，

A，B，C，D，E，F

しか使われていないのがわかります。これに数字の部分を追加すると，

0 1 2 3 4 5 6 7 8 9 A B C D E F

の**16種類の英数字**が使われていることがわかります。

次に**第1-3図**の4桁の英数字の1桁目のところをずっと追ってみてください。

0 1 2 3 4 5 6 7 8 9 A B C D E F

の順に繰り返されているのがわかります。どうやら英字の部分は，

A――10
B――11
C――12
D――13
E――14
F――15

のように対応しているような気がしませんか？

まさにその通りです。普通これらの英数字を

16進数

と呼んでいます（**第1-4図**）。

16進数	10進数
⓪	0
①	1
②	2
③	3
④	4
⑤	5
⑥	6
⑦	7
⑧	8
⑨	9
Ⓐ	10
Ⓑ	11
Ⓒ	12
Ⓓ	13
Ⓔ	14
Ⓕ	15

16進数は16個の英数字で構成される。

《第1－4図》16進数

16進数(hexadecimal notaion)

0～9，A～Fの16種の英数字で表記される数体系。

(例)	16進数	10進数
	1	1
	B	11

F　15
10　16

16進数については，付録の「10進数↔16進数変換表」を見て，その**数え方，表の見方**を良くつかんでおいてください。

3. マシン語のプログラムを入力したら

以上で16進数の大まかな概念をつかめたと思いますので，その知識を利用して，もう一度先の三つのプログラムを見比べてみてください。

第1-3図を省略した図→**第1-1図**
第1-1図を省略した図→**第1-2図**

であることがわかります。

雑誌等で良く見かけるのは，**第1-2図**のような形が多いと思います。**「マイコン誌」**等を見ると，楽しそうなゲームが載っていますね。特に面白そうなゲームがあると，たいていはマシン語のプログラムですね。すると**第1-2図**のようなマシン語リストがズラズラッと並んでいると思います。

あなたは，そのプログラムをあなたの**マシンに入力**することができますか？　それを**カセットにSAVE**したり，**LOAD**したり，**プログラムを走らせる**ことができますか？

さあ，だんだんと覚えていきましょう。まもなくあなたはどの形のマシン語のプログラムでも，自由に入出力できるようになることでしょう。

ここで最もポピュラーな形である**第1-3図**のプログラムを入力することを考えてみましょう。まずあなたのマシンの電源をONにしてください。

'OK'

の文字が表示され，カーソルが点滅していますね？

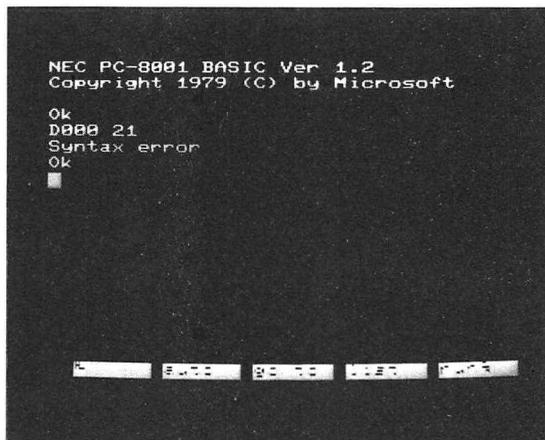
そこでまず第1行目を入力してください。

D000　21↗
(↗は**Return key**のことです)

とキー・インしましたか？　うまく入力できましたか？

ダメでしたね。Returnキーを押したとたん**ベルが鳴り，**

Syntax error

の文字が出たはずです（**写真1**）。

《写真1》Syntax errorがでた！

他の二つのプログラムについても同様の実験をしてみてください。うまくいきましたか？　やはりダメですか。これでは，せっかく雑誌に面白いプログラムが載っていてもキー・インして遊ぶことができませんね。

4. 二つのコマンド・レベル

どうしてダメなのでしょうか？　画面には，'OK'の文字が出ていたのに入力できないなんて。

実は，

'OK'の文字が出ているからこそ
マシン語のプログラムを入力できない

のです。というのは，

'OK'はBACICに対してOK

であって，**マシン語に対してはNO**の合図なのです。

ここでハッキリさせましょう。

あなたのマシンは，二つの状態を持っています。それは，

BASICを扱える状態
マシン語を扱える状態

の二つです。あなたのマシンは，今どちらの状態にあるかをあなたに知らせるために，**メッセージ**を送ってきます。それは，

OK――BASICが扱える状態
***――マシン語が扱える状態**

です（この辺の詳しい話は，私の「**PC-8000マシン語活用マニュアル**」(注)の第1章をみてください)。

今後便利なように，二つの言葉の定義をしておきます。

BASICのコマンド・レベル
＝'OK'の出ている状態
マシン語のコマンド・レベル
＝'*'の出ている状態

	メッセージ	使える言語	管理
BASIC・コマンド・レベル	OK	BASIC	BASICインタプリタ
マシン語・コマンド・レベル	*	マシン語	モニタ

《第1－5図》二つのコマンド・レベル

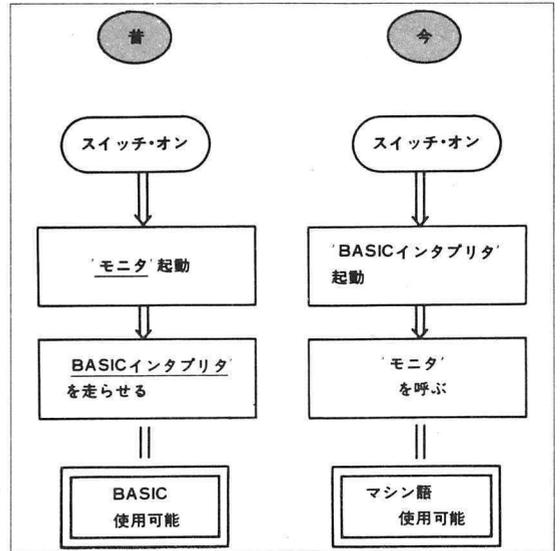
以上の話しから，
マシン語を扱うには，
マシン語のコマンド・レベル
にしなければならない事が理解できると思います。それでは，どうしたらあなたのマシンをマシン語のコマンド・レベルにすることができるでしょうか？

5. モニタを起動する

そこで予備知識として，**モニタ**という言葉を覚えてください。

> **モニタ（monitor）**
> コンピュータ稼動時において，最も基本となるメーカーのサポート・プログラム。入出力等の基本処理ルーチンを持っている。

もともとコンピュータは，マシン語が基礎ですから**モニタ**もマシン語の処理が基本となります。したがって本来は，電源ONの状態でモニタが起動し，マシン語が扱える状態になるわけです。ですから**BASIC**を使いたいと思えば，モニタを使って**BASICインタプリタ**というマシン語のプログラムをロードし，モニタを使ってそのプログラムを走らせるわけです。こうして初めてBASICが扱えるわけです。

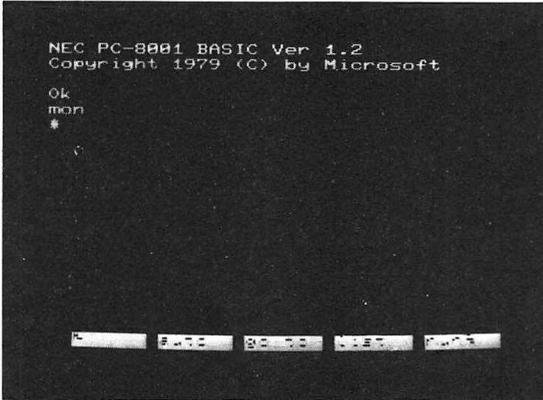
ところが昨今のハードウエア，ソフトウエアの進歩は，**コンピュータの操作性**を向上させてくれました。我々のマイコンもマシン語のみでなく，**高級言語であるBASIC**が標準装備され，プログラミングの生産性を向上させています。そのため**スイッチ・オン**でいきなりBACICが走れるようになってきました。そして，
マシン語とBASICとの
主従が逆転
し，モニタ・プログラムはむしろBASIC側から呼び出して使うという**従来とは逆の形**になってきているのです（**第1-6図**)。

《第1－6図》スイッチ・オンBASIC

そこで話をもとに戻します。
我々は，これから我々のマシンでマシン語を扱いたいと考えています。しかし，それには
マシン語のコマンド・レベル
にしなければならないことを知りました。それは，すなわちメーカー側が用意したサポート・プログラムである
モニタ
が稼動していなければならないということです。

> (注) 月刊「マイコン誌」(82年1月)別冊付録，
> マシン語の初歩の初歩から解説したもので，本書の入門版にあたります。お持ちの方は，ぜひお読みになってください。

さあ，そこで我々のマシンでもモニタを稼動させることに致しましょう。あなたのマシンは，今**'OK'**の文字が出ていますか？　OK？　それでは，

MON↗

とキー・インしてください。これで我々の**PC**もモニタが稼動します。**写真2**のように

《写真２》モニタが稼動し〝*〟が表示。

*

が表示されましたね。

6. 二つのコマンド・レベルを往復する

> **書式**：MON
> **目的**：内蔵のマシンランゲージモニタに制御を移す。
> **解説**：BASICモードから内蔵のマシンランゲージモニタにコントロールを移すためのステートメントです。マシンランゲージモニタの命令はマシンランゲージモニタのマニュアルを参照して下さい。コントロール―BでもとのBASICのモードにもどります。

これは，**「N-BASIC リファレンス マニュアル」**から抜粋したものです。つまり，

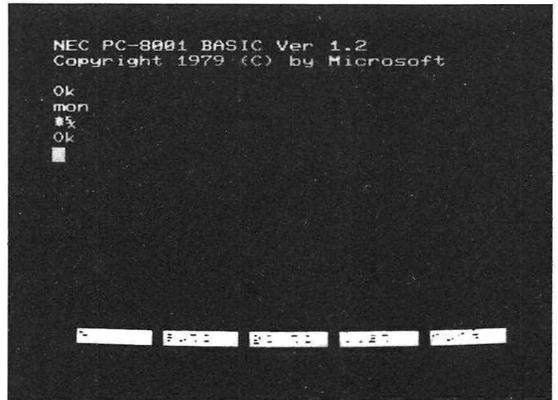
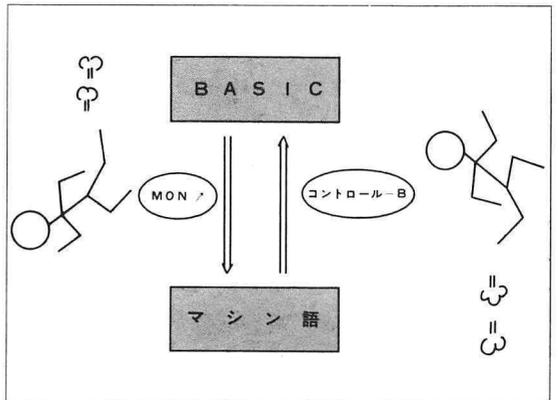
MONはN-BASICのコマンドの一つ

だったのです。

〈注〉本来モニタとは，効率的なコンピュータ利用を目的としたオペレーティング・システムですから，マシン語のサポートだけを目指したものではありません。そこでとくにマシン語のためのモニタを指すときは，**マシン語モニタ（マシンランゲージモニタ）**と呼びます。普通，マイコンでモニタと言う時は，マシン語モニタを指します。

さて，マニュアルでこのところを読むと，モニタからBASICモード（BASICのコマンド・レベル）への帰り方が書いてあります。

[CTRL] と [B]

のキーを同時に押してみてください（コントロール―B)。これでお馴染みのBASICモードに戻れました。(**写真3，第1-7図**)。

《写真３》モニタからBASICに戻す。

《第１－７図》二つのコマンド・レベルを往復する

これで我々は，

BASICのコマンド・レベル

マシン語コマンド・レベル

の**二つのコマンド・レベル間を自由に往来**できるようになりました。馴れるために，

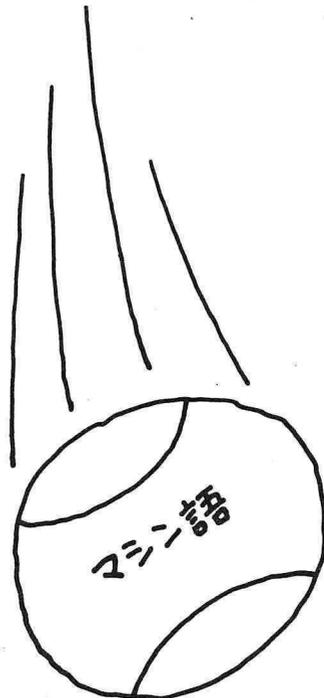
MON↗

コントロール―B

を繰り返し操作してみてください。もう，モニタなんて恐くありませんね。

<第1章のおわりに>

我々は三つのプログラムを観察することから始めて，いろいろな概念を理解し，我々のマシンの制御をモニタに移すことに成功しました。

次章では，いよいよ**モニタの扱い方**に触れていくことになります。BASICとは違った**新しいコマンド**が登場してきます。さあ，新しい世界に足を踏み入れていきましょう。

第2章
TM・Sコマンドの中身を探る

<はじめに>

本章から**モニタの使い方**を覚えていくことになります。そこには，BASICのコマンドとは違った素晴しいコマンドが展開されることでしょう。

本節では，モニタ・コマンドのうち

TMコマンド

Sコマンド

の二つが登場します。本章が終る頃には，あなたは**自由にマシン語のプログラムを入力できる**ようになっていることでしょう。なお，本章では合わせて，我々の理解可能な範囲で**モニタの内部構造**にも立ち入って行きます。楽しみにお読みください。

7. たった一つ，特殊なモニタ・コマンド

これからモニタ特有のコマンドの使い方を見ていくことになります。モニタのコマンドは，全部で8種類あります(第1-8図)。このうち，コントロール—Bは BASICに戻るためのコマンドですから，実際にモニタで使えるのは7種類となります。

コマンド	内　　容	コマンド	内　　容
S	セット　メモリ	W	ライト　テープ
D	ディスプレイ　メモリ	G	ジャンプ
L	ロード　テープ	↑B	リターン　ベーシック
LV	ロード　ベリファイ　テープ	TM	テスト　メモリ

《第1－8図》モニタのコマンド

前章において，モニタはマシン語を扱うために必要なことを知りました。そうすると，この7種類は**マシン語を扱うために必要**ということになります。

ところが，ところが——。

モニタ・コマンドの7種類のうち，たった一つだけ**マシン語に関係のないコマンド**があります。それはあなたのマシンのために便利な——否，必要不可欠な機能を実現するためにあります。仮にあなたが**BASICしか使わない**，あるいはもっと極端に**市販のプログラムを買って使うだけ**だとしても，あなたのマシンのためにこのコマンドの使い方だけは覚えておいた方が良いでしょう。それは，あなたのマシンの

健康診断をする

ためのコマンドです(第1-9図)。

《第1－9図》モニタ・コマンドの用途

8. すべての人のために——TMコマンド

それは，

TM：テスト　メモリ

コマンドです。

たぶん，あなたは**メモリ**という言葉を知っていると思います。知らなくても結構，メモリについては次ブロックで触れます。

あなたのマシンは，**コンピュータ**と呼ばれます。コンピュータはその昔，**'電子頭脳'**とも呼ばれました。「頭脳」というからには，**物事を記憶する**ことができ

るわけです。あなたのマシンも，当然プログラムやデータを記憶できます。このコンピュータの記憶を受け持つ装置を

メモリ

と呼んでいます。

あなたのマシンにもメモリがついていて，たくさんの記憶場所を持っています。あなたのマシンは何**K**システムですか？

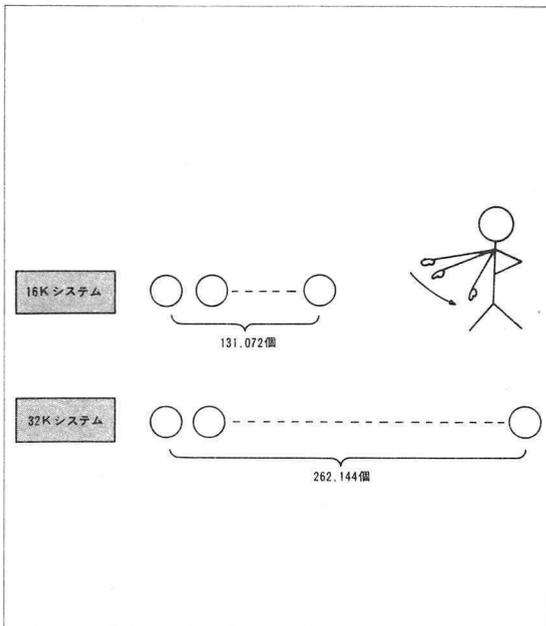
もしあなたのマシンが16Kシステムなら

131,072

もの記憶場所を持っています。もし32Kシステムならなんと，

262,144

個所もの記憶場所を持っているのです。これは膨大な数ですね(**第1-10図**)。

あなたのマシンは，記憶装置としてこんなにもたくさんのメモリを持っているのです。こんなにたくさんあって，果して**全てうまく動いている**でしょうか？どこか故障していないでしょうか？

《第1－10図》膨大な数の記憶場所

あなたは，それを確かめたことがありますか？

えっ？ そんな膨大な数のメモリをいちいちチェックできない？

イヤ，できるのです。そのためにモニタに

TM

コマンドがあるのです。

9. 実験——TMコマンドを使う

TMコマンド
目的：メモリをテストする
書式：TM↗

TMコマンドは，あなたのメモリの良否をチェックしてくれます。もし不具合があれば，どのメモリのどの辺が悪いかも教えてくれます。したがって，非常に重要なコマンドであると言えます。

○まだメモリ・テストを行ったことのない人

○メモリを増設した人

(とくにそのメモリが純正品でない時，メモリチェックは重要ですね)

ぜひ，一度TMコマンドを実行してみることをお勧めします。

TMコマンドの使い方は簡単です。

あなたのマシンを，マシン語のコマンド・レベルにしてください。'*'が出ていますね。そうしたらおもむろに

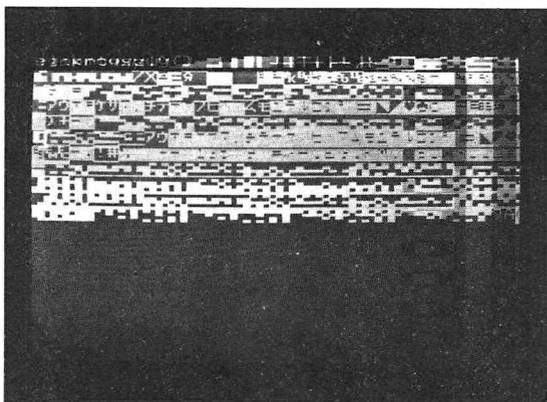
TM↗

とキー・インしてください。モニタが，あなたのメモリのチェックを始めます。この間，どんなことが起こるかを順に書いておきます。

①RETURNキーを押した直後

左端の'*'が点滅を開始します。

②5.5秒経過後

画面の最下端が，乱れ始めます。これでびっくりしないでください。メモリ・チェックは順調に進んでいます。その後も画面のあちこちが乱れます。(写真4)。

《写真4》TSコマンドでメモリ・チェック

(どうしてこういう現象が起こるのか，またTMコマンドの中身を知りたい人は，次節を参照ください)。

③約21秒後

画面の乱れが止まり，すべての表示が消えてしまいます。先程の'*'の位置でカーソルが点滅しているだけです。メモリ・チェックは順調に進んでいます。もうしばらくお待ちください。

④約3分30秒後

(16Kシステムの人は，約1分45秒後)

画面はパッと電源ONの状態となり，処理の終了となります。

以上の経過を順調にたどった場合，あなたのマシンのメモリは正常です。もし不幸にして

途中でスピーカーが鳴りっぱなし

になった時，残念ですがあなたのマシンのメモリは不良です。この時の処置の仕方は，まだまだ我々の知識では不足ですから，あとのところで診断の仕方を説明します。スピーカーが鳴っていますが，BASICのようにSTOPキーを押しても止まりません。とりあえずマシンの電源を切り，もう少し先まで本書を読み進めてください。

なお，チェックが正常に終了した場合でも，後日あなたのメモリがいかれた時（そういうことがないようにお祈りしていますが）困らないように，

わざとメモリ・エラーを起こす方法

をあとで紹介する予定でいます。ご期待ください。

――備えあれば憂いなし，ハイ。

10. TMコマンドの中身

本節では，モニタがTM処理をするのにどんなことをしているのか，

モニタの中身

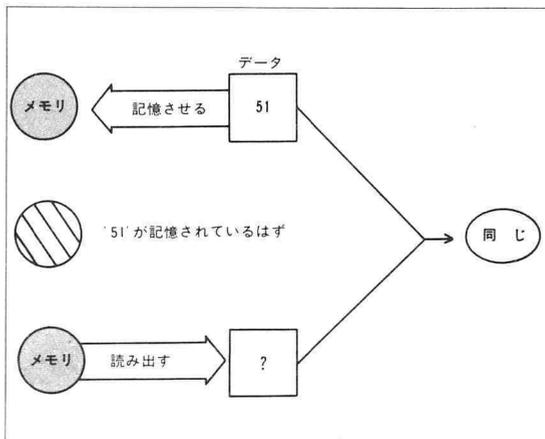
を少し覗いてみることにします。本文の流れとは無関係ですから興味のない人は，読み飛ばしてください。

一つのメモリについてのテスト処理の基本は，次のとおりです。

そのメモリにあるデータを記憶させる

そのメモリから記憶内容を取り出す

最初に記憶させた値＝今読み出した値？

これが等しければ，**そのメモリは正常**であると考えられます。また，もし異なるならば**そのメモリは異常**であると判定されます(第1-11図)。

《第1－11図》メモリをチェックする

以上が一つのメモリについての処理です。これを全メモリについてチェックするわけですが，その流れは次の通りです。

① モニタが**コマンドの1字目**として'T'をキャッチすると，続いて

M↗

と入力されるかチェックします。もしそうであれば②以下の「テスト・メモリ処理」に移ります。もしそうでなければ，'?'を表示して**コマンド・エラー**であることを示します。

② メモリ・チェックは256ずつまとめて処理します。また，処理の順序はメモリのうしろの方からチェックして行き，256ずつ前の方へと進めて行きます。

③ **1グループ(256個)内の処理**

1) メモリの前の方から順に

0，1，2，……，255

のデータを記憶させる

2) メモリの前の方から順に記憶内容を取り出し，

0，1，2，……，255

と比較する。もし比較の途中で異常なメモリが現われれば，ただちにそこで処理をやめ，⑤の処理に移る。

④ ③の処理を行った結果すべてのメモリに異常が認められなかった時は，プログラムの流れをBASICインタプリタの先頭にセットします（マシン語の命令で書けば，

ＪＰ　０００ＯＨ

となります)。すると画面上には電源ＯＮと同じ状態が現われます。

⑤ もしメモリに異常があれば

読み込んだデータ

記憶させたデータ

エラーの起きたメモリ

の三つをあなたに知らせる準備をし，内蔵スピーカーのスイッチをＯＮにし，プログラムの活動を停止します（マシン語で言えば，

ＨＡＬＴ

という命令のことです)。

以上がＴＭコマンドの中身です。いずれあなたがマシン語を読めるようになった時，一度ＴＭコマンドのプログラムを解析してみてください。いま述べてきたことがもっと克明にわかって面白いと思います。ＴＭコマンドのスタート番地は，５ＤＥ６番地です。

なお蛇足ながら，我々のマシンはテレビ画面を表示したり，その表示方法を指定したりするにもメモリを使っています。メモリのチェックを始めて約20秒後には，この部分のチェックに入ります。ＴＭコマンドを実行中，画面が乱れたのはこのためです。

11. Sコマンドの起動

メモリのチェックが終ったところで，今度はいよいよ**マシン語関係**のコマンドに進みましょう。まずその入り口は，

マシン語を入力する

ことから始めます。使用するコマンドは，

Ｓコマンド

です。

> **Sコマンド**
> **目的**：メモリの内容を変更する
> **書式1**：**S 〈アドレス〉** ↗
> **書式2**：**S**↗

Ｓコマンドについても，実験を通して身につけていくことに致します。例として**第1-12図**のプログラムを入力することにします。

番　　地	マシン語
D000	21
D001	38
D002	18
D003	CD
D004	ED
D005	52
D006	C3
D007	66
D008	5C

《第1－12図》Sコマンド実験プログラム

４桁の英数字が番地，２桁の英数字がマシン語を表わすことは，第１章で触れました。そこでそのつけ足しとしてもう少し感覚的に説明しておくと，

番　地：記憶場所の位置

マシン語：命令やデータ

ということになります。これを**第1-12図**のプログラムで見ると，たとえば１行目は，

D000番地という記憶場所に

21というマシン語を記憶させる

という意味です。

そこでSコマンドの使い方ですが

S＋〈最初の番地〉＋↗

が基本です。たとえば**第1-12図**の場合でしたら最初の番地は，D000ですから

SD000↗

と入力するわけです。これでSコマンドが起動されます。**写真5**が，その様子を表わしています。

《写真５》Sコマンド起動

12. マシン語の入力

さあ，Sコマンドが動き始めました。さっそくプログラムを入力してみましょう。最初のマシン語は21ですから，

21

とキーインしてください。**写真6**のようになるはずです。表示の見方は**第1-13図**の通りです。すなわち

一の左側：変更前のデータ

一の右側：変更後のデータ

というわけです。なお，'一'の左側の値は，各自のマシンのその時の状態によって異なります。

《写真６》マシン語のプログラム入力

《第１－13図》Sコマンドの見方

入力の仕方がわかったところで，**第1-12図**のプログラムをどんどん入力していってください。この時，

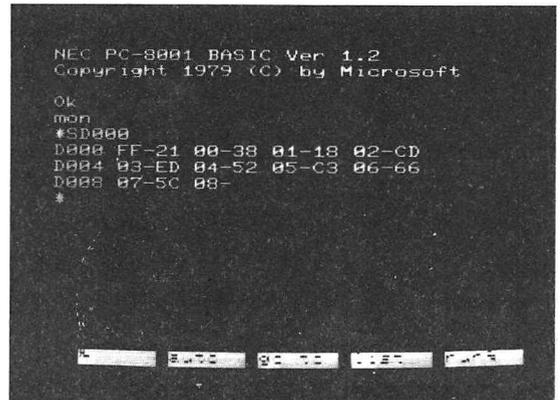
１行40字モード→４データ

１行80字モード→８データ

入力すると改行され，その時の番地が表示されます。すべてのデータの入力が終りましたら，

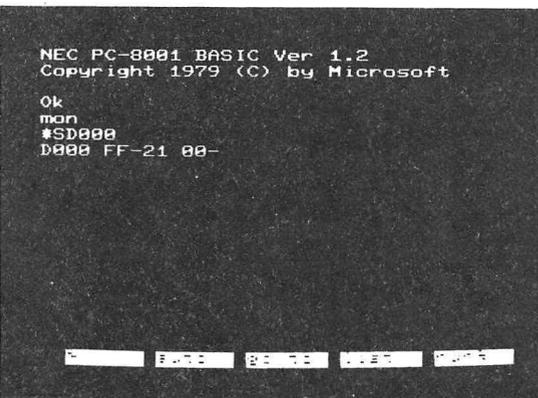
RETURN KEY

STOP KEY

のいずれかを押してください。すると'*'が表示されマシン語のコマンド・レベルに戻ります(**写真7**)。これはさしずめBASICでプログラムの入力が終り，'OK'が表示されコマンド待ちになったのと同じことです。

《写真７》マシン語のコマンド・レベルに戻る

13. 便利な機能

以上がSコマンドの基本です。しかし，Sコマンドには他にも便利な使い方がありますので，それを次にまとめておきます。

①変更の必要のないとき

SPACEキーを押せば，次の番地に進むことができます。

②一つ手前の番地に戻りたいとき

DELキーを押すか，コントロール-Hで戻れます。もちろん**オート・リピート**も効きますから，キーを押しっぱなしにすれば，番地はどんどん手前に戻ります。

③入力の途中で'*'に戻ったとき

これは次のような場合が考えられます。

- 16進数でない数を入力したとき。たとえばSとか。
- マシン語を1桁しか入れないでRETキーを押したとき。
- 入力の途中なのにRETキーを押してしまったとき。

こんな時は非情にも，'*'が表示され，マシン語のコマンド・レベルに戻ってしまいます。するとまた，

S××××↗

と押すことになります。しかし，これは面倒ですね？　こんな時は，単に

S↗

とだけ入力してください。これについては，節を改めて説明します。

以上3項目について，よく御自分で実験してみてください。すぐに慣れると思います。

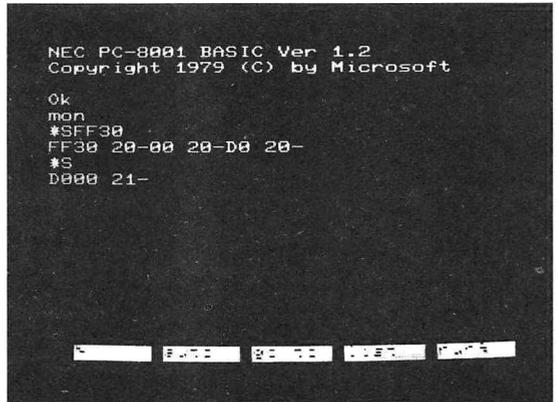
14. アドレスのメモリカウンター

この節も**モニタの中身**の話しです。したがって，興味の無い人は読み飛ばしても大勢に影響ありません。

それでは，次の実験をしてみてください。

〈実　験〉

① マシンの電源を入れる。
② マシン語のコマンド・レベルにする。
③ SFF30↗
④ 00　D0　とキー・イン
⑤ STOPキーを押す
⑥ S↗

やり方，わかりました？　実験がうまくいけば，D000番地からSコマンドが働くはずです（**写真8**）。

《写真8》D000番地からSコマンドが動く

なお念のために申し上げますが，この節の話しは初めての方には少し難しいかもしれません。ですから無理にわかろうとしないで（なにしろわかる必要は全くありませんから），本当に次節まで読み飛ばした方が得策です。

さて，この実験がうまくいったなら，もう一度STOPキーを押して'*'に戻してください。そして，④のデータを次のように変えて実験を繰り返してください。

00　D1
00　D2
00　E0
23　E1

結果はどうでしたか？　鋭いあなたは，どうやら

FF30番地
FF31番地

が**Sコマンドの〈最初の番地〉**に関係ありそうなことに気が付かれたのではないでしょうか？　実験の結果をまとめると，次のようになります。

④の入力データ		Sコマンドの最初の番地
00　D1	→	D100番地
00　D2	→	D200番地
00　E0	→	E000番地
23　E1	→	E123番地

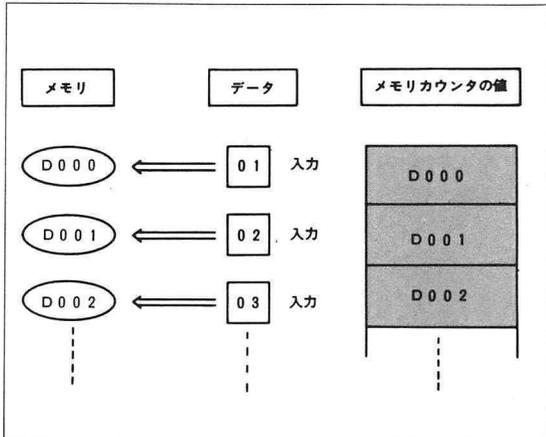
実は，この辺のしくみは次のようになっています。

> **アドレスのメモリカウンター**
> 　モニタは，Ｓコマンド実行中，常にアドレスをメモリカウンターに記憶させている。**メモリカウンター**の番地は次の通り。
> 　　ＦＦ３０番地：番地の下位２桁
> 　　ＦＦ３１番地：番地の上位２桁

アドレスとは**番地**のことです。また，

　　アドレスのメモリカウンター

　　　　＝ＦＦ30～ＦＦ31番地

のことです。あなたがＳコマンドを実行してマシン語を入力している時，モニタ内部ではその都度，いま入力した番地を**メモリカウンター**に記憶させています。つまり１データ入力するごとに，メモリカウンターの値は変わります(第1-14図)。

《第１－14図》メモリカウンターの様子

ところでＳコマンドを開始するとき，**〈最初の番地〉を省略**して単に

　　　Ｓ↗

とすると，**〈最初の番地〉としてアドレスのメモリカウンターに記憶されている番地が採用**されます。前節の「Ｓコマンドの便利な機能③」は，このことを利用したものです。

マシン語を入力すると，メモリカウンターの値が変わることはわかりましたが，逆に

　　メモリカウンタにデータを強制的に入力

させ，本当にその値がセットされたかを見てみようとしたのが先の実験です。

いかがですか？　面白かったでしょう？

> **〈第２章のおわりに〉**
> 　本章で，ついにマシン語の入力に成功しました。しかし，それだけでは悲しいかな何もできません。あなたは
> 　うまく入力したか，チェックできますか？
> 　そのマシン語を走らせられますか？
> まだまだ知りたいことがたくさんありますね。
> さあ第３章以下が，あなたをお待ちしています。
> 元気に学習を続けてください。

第3章 マシン語プログラムGO↗

〈はじめに〉

本章の終了の頃には，あなたは，

他人様の作ったマシン語のプログラム

で遊べるようになっているでしょう。とにかく本章の最終目標は，マシン語のプログラムを走らせてみることですから。

15. プログラムを見る

あなたがBASICでプログラムを作った時，まずマシンに入れますね。そのあと，どうします？

LISTを取る——プログラムを見る

RUNさせる——プログラムを走らせる

の二つに別れるでしょう。

マシン語の場合も同じです。プログラムを入力（Sコマンド）したあとは，その時と場合に応じて

プログラムを見る————Dコマンド

プログラムを走らせる——Gコマンド

の二つのケースに分れます。

ここでは，先にDコマンドから見ていくことにします。そして，前章で入力した**第1-12図**のプログラムがうまく入力されたか確認してみることにします。

Dコマンド

目的：メモリの内容を表示させる

書式1：D〈アドレス〉↗

データを16個表示する

書式2：D〈AAAA，BBBB〉↗

AAAA～BBBB番地のデータを表示する。

それでは，ささっそくDコマンドを使ってみましょう。**第1-12図**は入力しましたね。

DD000↗

とキー・インしてください。**写真9**のように表示されるはずです。

```
NEC PC-8001 BASIC Ver 1.2
Copyright 1979 (C) by Microsoft

Ok
mon
*DD000
D000 21 38 18 CD ED 52 C3 66
D008 5C 08 09 0A 0B 0C 0D 0E
*
```

《写真9》Dコマンドをキー・イン

16. Dコマンドの便利な機能

うまく表示されましたら，データの数を数えてみてください。16個ありますね。

DD000↗

とすると，**D000番地から16個分のデータが表示される**のです。

それでは**第1-12図**のプログラムと照らし合わせてみてください。もっとも**第1-12図**のプログラムは，データが9個しかありませんから，少し余分に表示されています。しかし，余計に表示される分には困りませんから，気にしないでおくことにしましょう。

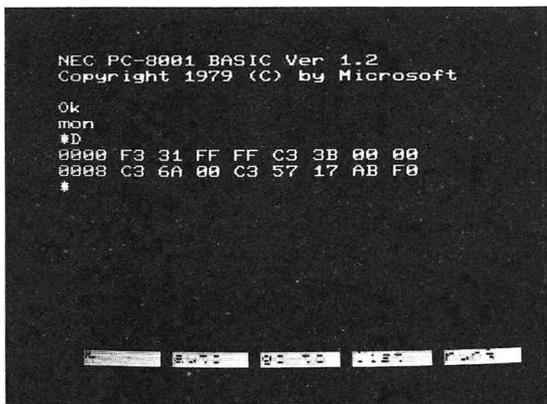
以上がDコマンドの使い方の基本です。もちろん他にも使い方はあります。それを次に列挙しておきます。

① Sコマンドの時のように，番地を省略するとどうなるでしょうか？

D↗

とキー・インしてください(**写真10**)。予想は裏切られましたね。**Dコマンドでは，アドレスのメモリカウンターが設けられていません。**番地を省略すると，

番　地＝0000

と見なされ，0000番地から16個分のデータが表示されます。

《写真10》番地を省略しD↗とキー・イン

② 長いプログラムを見たい時，もし1回に16個のデータしか表示されないとしたら，何回も**Dコマンド**をキーインしなければならず，不便だと思いませんか？　そんな時は次のようにキーインしてください。

D〈最初の番地〉,〈終りの番地〉↗

こうすると，

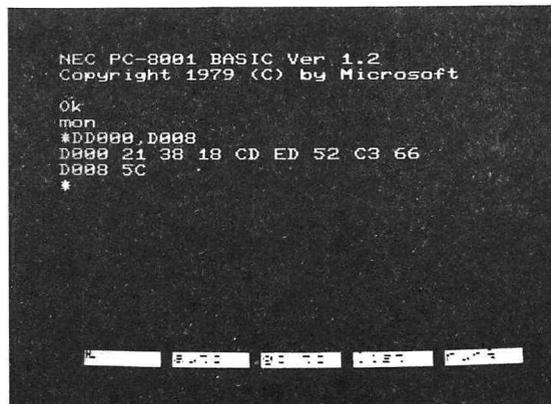
〈最初の番地〉〜〈終りの番地〉

までのデータが表示されます。

例を二つあげましょう。

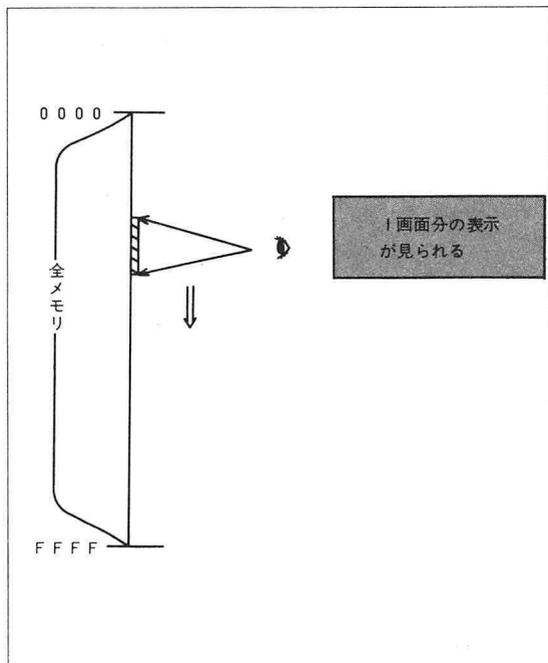
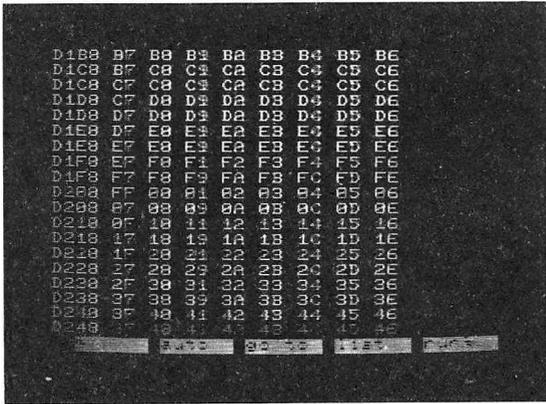
まず**〈その1〉**。**第1-12図**のプログラムを見る場合でしたら，

DD000，D008↗

でOKです。**写真11**のように表示されますね。

《写真11》第1－12図のプログラムを表示

〈その2〉。ついでですからあなたのマシンの**全メモリ**の様子を見てみましょう。

D0000，FFFF↗

でOKです。記憶内容がメモリの最初の方から順に表示されていきます(**写真12，第1-15図**)。

《第1－15図》全メモリの様子を見る

《写真12》マシンの全メモリを見る

ところで――。

これではあまりに高速で表示されていくため，途中のデータを十分にチェックできませんね。そこで

表示を止めたい時――ESCキー

を押してください。また

表示を開始する時――ESCキー

で動き始めます。もし，

途中でやめたい時――STOPキー

を押してください。再び'*'の状態に戻ります。

以上**Dコマンド**についていろいろ見てきました。納得がいくまで自分で実験してみてください。自由にDコマンドを使いこなすことができるようになると思います。

17. プログラムを走らせる

次からいよいよマシン語のプログラムを走らせることになります。使用コマンドは，

Gコマンド

です。イザ。

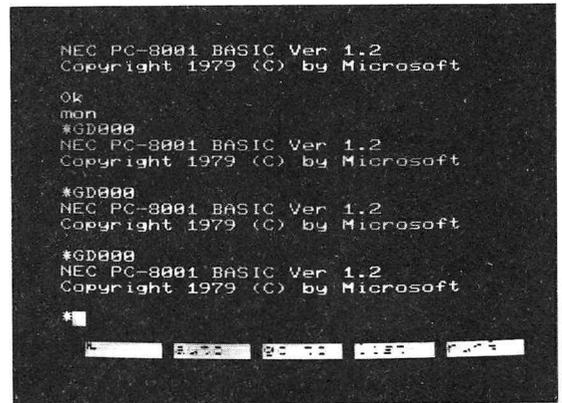
> **Gコマンド**
> **目的**：プログラム・カウンタを指定の番地にセットする。
> **書式**：G<ジャンプ・アドレス>↗

プログラム・カウンタを指定の番地にセットする'――難しいことが書いてありますね。この枠内の意味は，後日理解していただくとして，ここでは具体的に説明していくことにします。

そこで例としていままで何度も例に使ったお待ちかね**第1-12図**のプログラムを実際に走らせてみることに致します。**第1-12図**のプログラムは，D000番地から始まっています。そこで，

GD000↗

とキーインしてください。これで**第1-12図**のプログラムが走りました。短いプログラムですから，すぐ処理が終わります。何が起こりましたか？

良くわからなければ，もう一度走らせてみてください。それでもわからなければ，さらにもう1回走らせてください。**写真13**は，3回走らせた様子を表わしています。

《写真13》3回走らせた様子

第1-12図のプログラムは，**N-BASICの初期画面を表示する**ものです。したがって3回走らせれば，メッセージが3回表示されるわけです。この節は，プログラムの解析が目的ではありませんから，**第1-12図**のプログラムの説明はこの位でやめておきます。

この節で知っていただきたいことは，要するに

マシン語のプログラムを走らすには

G<プログラムの先頭番地>

ということです。

本節の最後に，面白いプログラムを走らせてみましょう。長い長いマシン語のプログラムです。何と24Kもあります(Kについては後述)。それは，

N-BASICインタプリタ

というプログラムです。

G↗

とキーインしてください。電源ONと同じ状態になる

でしょう（正確には，メモリの一部が異なりますが)。なぜなら，N-BASICインタプリタのスタート番地は，００００番地であり，Gコマンドで

〈スタート番地〉を省略

→〈スタート番地〉＝００００番地

となるからです。これは，Dコマンドと同じですね。

〈第３章のおわりに〉

本章，いかがだったでしょうか？

ついに我々は，

マシン語のプログラムを走らせる

ことに成功しました。ということは，

マシン語を入力し，

マシン語を走らせる

ことが可能になったのです。必要とあらば，その**マシン語を見る**こともできます。

――だからといって，さっそく雑誌に載っているような長～いマシン語のプログラムを入力し，走らせてみようなんてせっかちなことはやめた方が良いでしょう。なぜなら，仮にその入力にミスがなく，プログラムが動いたとします。それがゲームなら，遊ぶことができるでしょう。しかし，ゲームにも飽き，やめたあとどうしますか？――電源を切る。切ったらいま入力したプログラムは消えてしまいますよ！

我々が知らなくてはならないことは，まだまだたくさんあります。たとえば

マシン語のプログラムをSAVE

すること位は知りたいですね。次章のターゲットはその辺にあります。

第4章 カセット・オペレーションとメモリ・エラー

〈はじめに〉

いよいよ第1ブロック，最後の章に突入します。本章の中心は，残されたモニタ・コマンド——カセットのオペレーションになります。と同時に，本章の最後では「非常に変わった実験」を行います。それはおそらくPCのユーザーはあまり経験したことのない実験だと思います。もちろん,我々の知識だけで理解できます。張りきってまいりましょう。

18. 頑張れ，カセット！

これからカセット・テープ・レコーダーとの入出力を見ていくわけですが，

か・せ・っ・と

と言っても馬鹿にしないでください。現在，**マイコン**の外部記憶装置として**ミニ・フロッピー・ディスク**がだいぶ普及してきました。このため，とかくカセットは低く見られがちです。

しかし，**外部記憶装置としてカセットを使う**——というアイデアがなかったなら，これ程速いペースでの

マイコンの普及

マイコンの大衆化

はあり得なかったと思われます。カセット・レコーダーの果たした役割は数あるマイコン周辺機器の中ではおそらく**トップ**であるだろうし，それは

ＲＦモジュレーター

７セグメント・ディスプレイ

以上だろうと思われます。

さらにカセットとミニ・フロッピーを併存させても，それぞれにハード上の長所があり，カセットがまったく無視されるということはありえないでしょう。カセットの持つ，

高信頼性

耐久性の強さ

保存のしやすさ

は，まさにフロッピーのバック・アップとして最適であり，我々としてはカセット，フロッピー相互の利点を利用するのが得策でしょう。

たとえばオーディオにおけるプレーヤーとカセット・デッキを見れば，その操作性の良し悪しは一目瞭然であるし，これを大型コンピュータと対比させれば，

磁気テープ・ユニット↔カセット

ディスク↔ミニ・フロッピー

ということになるでしょう。

19. 録音のオペレーション

BASICにおけるカセット・レコーダーへのオペレーションは，大別して２種類あったと思います。それは，

プログラムの保存に関するもの

データの保存に関するもの

の２種類でしたね。

マシン語においても，当然

プログラム

データ

の２種類を入出力するわけですが，その方法は

まったく同じ

になります。ここらあたりが，マシン語の原始的なところでしょう。なぜならマシン語においては，

プログラム＝××××〜××××番地

データ＝××××〜××××番地

という形態を取り，**形の上からは，両者はまったく同じスタイル**をしています。このため，カセットのオペレーションにおいても，

××××〜××××番地を保存

という形式で，両者は**区別なく**扱えます。

Wコマンド

目的：メモリの内容をオーディオ・カセットに出力する。

書式：W〈AAAA，BBBB〉↗
AAAA～BBBB番地の内容が，オーディオ・カセットに出力される。

マシン語のプログラムやデータ内容をオーディオ・カセットに録音するには，

Wコマンド

を用います。書き方は，Dコマンドの書式２に似ています。

それでは再び**第1-12図**のプログラムを例に実験してみることに致しましょう。

スタート番地＝D000番地
エンド番地＝D008番地

です。したがって，録音の手順は次のようになるでしょう。

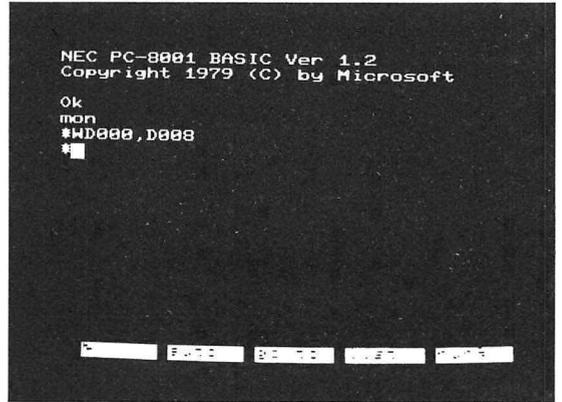
①カセットをレコーダーにセットする。
②キー・ボードから
WD000，D008
と入力する（**まだRETURNキーは押さない**）。
③カセット・レコーダーの録音ボタンを押し，テープの走行が安定するのを待つ。
④RETURNキーを押す。

以上の手続きで録音が開始されます。しばらくお待ちください。ここから二つのケースに分れます。

正常終了――“＊”コマンド待ち
異常終了――“？”が表示されたあと
“＊”コマンド待ち

もし録音に失敗したら，もう１度やり直してみてください。それでもダメなら，テープを取り換えてみてください。それでもダメなら，カセット・レコーダーを取り替えてみましょう。それでもダメなら，あっさり締めましょう。

20. ベリファイの原理

さあ，録音はうまくいきましたか？ **写真14**に，無事に録音が終了した場合を示します。

《写真14》無事に録音完了

続いて録音がうまくいったか，ベリファイを取ってみます。ところで，**ベリファイの意味**が良くわかっていない人がいるかもわかりませんので，次に簡単に説明しておきます。

verify〔vérifâi〕vt.
〈対照・調査して事実を〉確かめる「新英和中辞典」（研究社刊）

第1-16図をみてください。

左側の**‘メモリ’**というのが，あなたのマシンの記憶装置です。現在，D000番地から**第1-12図**のプログラムが記憶されています。その内容をカセットに録音したものが，右側の図です。録音が正常に行われれば

メモリの内容＝録音内容

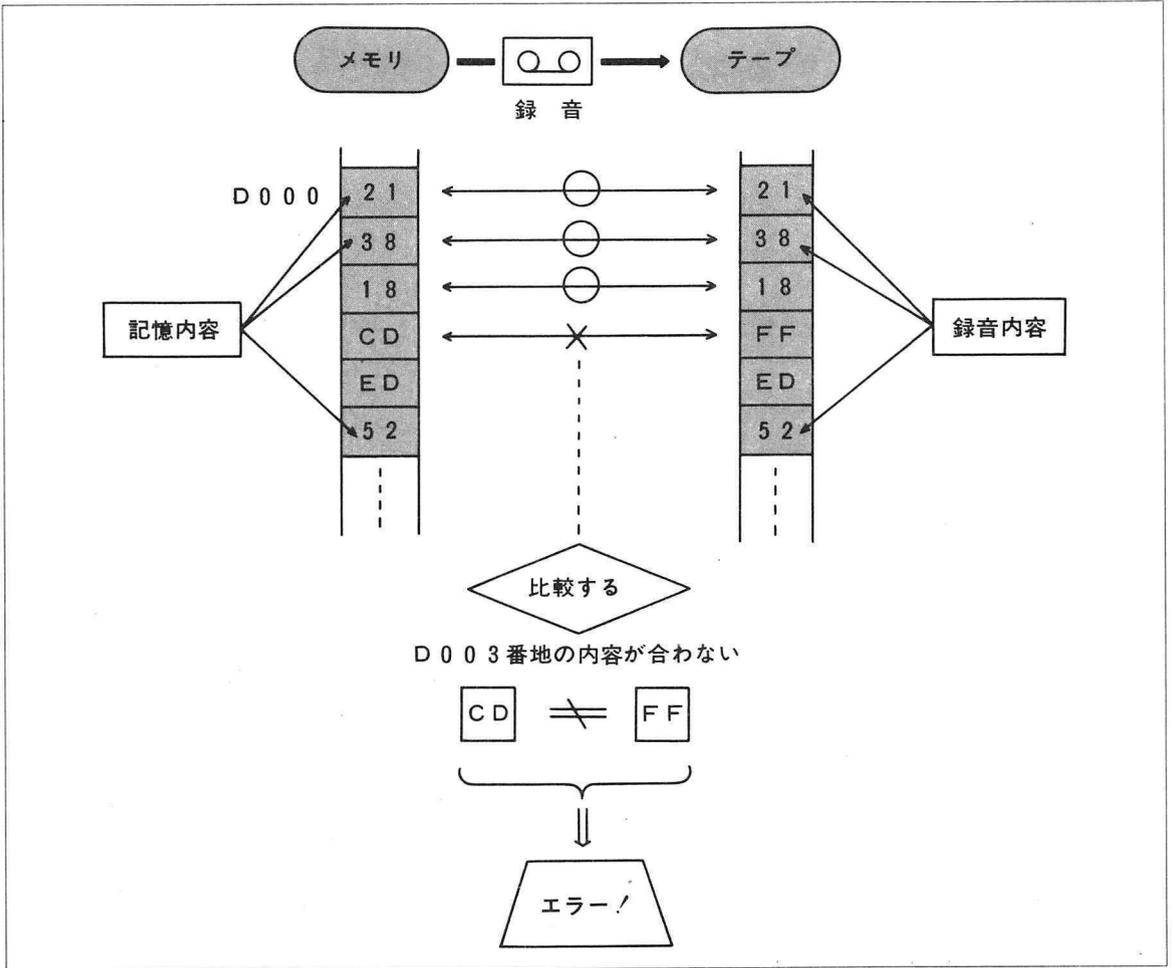
になるわけです。図では，たまたまD003番地の録音に失敗したため，

D000番地の記憶内容：CD
≠D000番地の録音内容：FF

となっています。

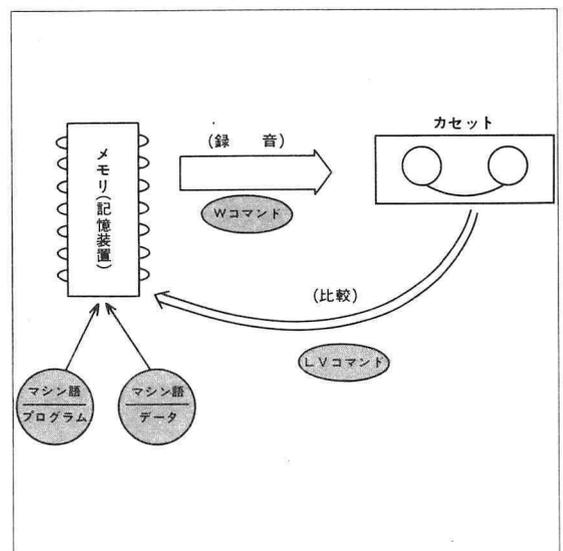
この状態でベリファイを取ったとします（たとえばBASICのプログラムでしたら，

CLOAD？↗

とやりましたね。マシン語でのやり方は，あとで説明致します）。ベリファイが開始されると，あなたのマシンは録音された内容とメモリの内容を一つずつチェックしていきます。録音された全データをチェックし，すべてのデータがOKであれば，

《第１－16図》ベリファイの機能

録音ＯＫ！
を表示しますし，途中に１ヶ所でも異なるデータがあれば，直ちにベリファイ処理を中断し，
録音失敗！
を知らせてきます。

21. ベリファイのオペレーション

以上がベリファイの原理です。そこで我々も，先程録音したテープのベリファイを取ってみることにします(第1-17図)。

《第１－17図》マシン語のプログラム・データの録音手順

LVコマンド
目的：テープに書かれている内容とメモリの内容を比較する。
書式：ＬＶ↗

マシン語のプログラムのベリファイを取るのは，簡単です。その手順を次に示します。

①先程のカセットを，録音開始位置まで巻き戻す。
②キー・ボードから

ＬＶ↗

とキー・インする（この時，RETURNキーを押してもかまいません。
③カセット・レコーダーの再生ボタンを押す。
④RETURNキーを押す。

以上の手順でベリファイ処理が開始されます。ここからWコマンド同様に二つのケースに分れます。

正常終了──'＊'コマンド待ち
異常終了──'？'が表示されたあと，
'＊'コマンド待ち

22. わざとエラーを出す

ベリファイ──うまくいきましたか？

ただうまくいったからといって，このまま先に進んでも面白くありません。もう一つ実験してみましょう。今度はわざと**失敗**してみます。

次の２点を確認してください。

①マシンに**第1-12図**のプログラムが正しく入力されていること（Dコマンドを使ってください)。
②カセットにそのプログラムが正しく録音されていること（ＬＶで正しいと確認)。

ＯＫですか？

続いてメモリのＤ００３番地の内容を

ＣＤ→ＦＦ

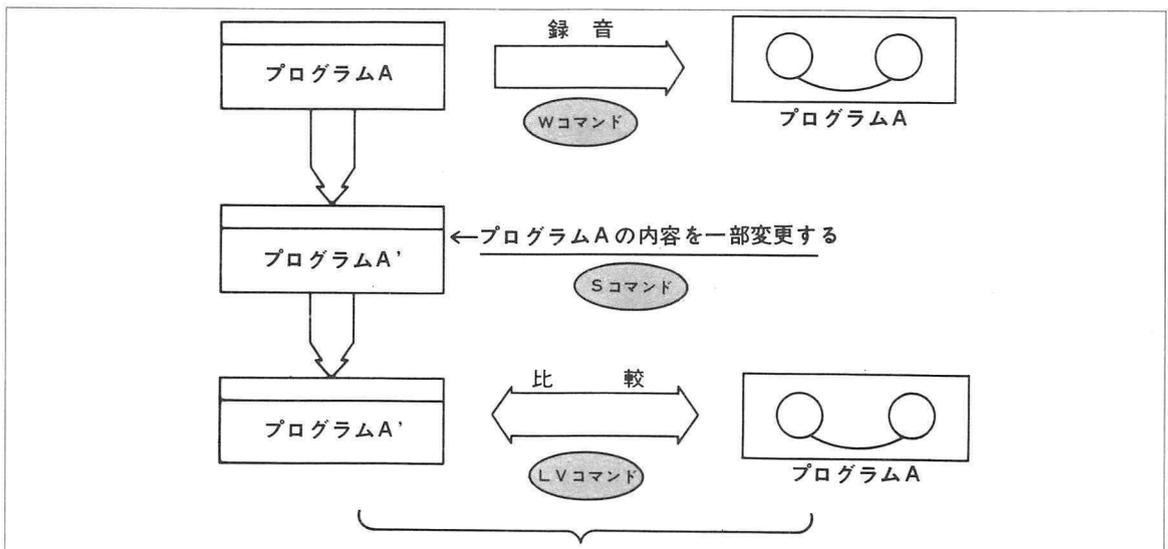
に変えてください。Sコマンドを用いればできますね。

ＳＤ００３↗
ＦＦ
STOP (or RETURN)

でＯＫです。これでメモリの内容とカセットの録音内容とに相違点ができました。カセットを巻き戻し，もう１度

ＬＶ↗

でベリファイを取ってください。

どうですか？　今度は明らかにエラーとなりましたね(**第1-18図**)。

《第１－18図》LVコマンドでエラーを作る！

23. 最後のモニタ・コマンド

さあ，カセットのオペレーションの最後になります

Lコマンド

に入りましょう。そしてこれがモニタ・コマンドの最後になります。

> **Lコマンド**
> **目的**：テープに録音されているデータをメモリ内に読み込む。
> **書式**：L↗

Lコマンドの使い方は，LVコマンドとまったく同じですから簡単です。次の手順により，自分で実験してみてください。

①マシンの電源を切る。
　これでメモリの記憶内容は全て消えてしまいます。
②先程のテープを巻き戻す。
③マシンの電源をオンにし，マシン語のコマンド・レベルにしてからLとキー・インする。
④再生ボタンを押す。
⑤RET キーを押す。
⑥テープ・リード・エラー（'?'）なら②からやり直す。
⑦GD000↗

以上の手順により，

N-BASIC スタート・メッセージ

が表示されれば，あなたのカセット・レコーダー・オペレーションは全て完璧に終了したはずです。

24. TMコマンド・エラーへのお誘い

前節を持ちまして，遂に我々は，

全てのモニタ・コマンドを突破！

したことになります。すなわち，我々は**マシン語をあやつる**(その中身は別として)ことに関しては全てを理解したことになります。さあ，そこでいよいよ第2章でお約束した

TMコマンドでエラーを発生させる実験

をここで試みることにします。あなたは既にそれを理解するのに十分な知識をものにしていますから…。

TMコマンドにおけるエラー――これを経験したことのあるPCのオーナーは，おそらくほとんどいないのではないでしょうか？　というのはICの高信頼性は目を見張るものがあり，メモリの不具合にぶつかる人はよっぽど運の悪い人だと思われます。もしかすると，それは**“ジャンボ宝クジ”に当る**より難しいかもしれません。そんなわけで，大部分のオーナーはTMコマンドを実行しても，難なくエラー無しの状態を体験すると思います。しかし，しかし――。これから我我は，あまり他人の味わったことの無い

TMコマンド・エラー

をこれから体験することができるのです。

ところで，我々は第2章でTMコマンドの原理を知りました。それによると

メモリに記憶させた内容
≠
そのメモリから読み出した内容

のとき,TMコマンド・エラーが発生する'ということでした。とすれば，我々がこれからそれを体験しようとすれば，

自分のマシンのメモリを壊さねばならない

ことになります。それは困りますね。

ご安心ください。これから私が紹介しようとする方法は，

ハードを壊す

のではなく，

ソフトにより強制的にエラーを発生させる！

方法です。あなたのメモリは，何ら傷むことはありません。しかも，すでにあなたはそのプログラムにお目にかかっています。

25. メモリ・エラー発生

注意深い読者は，それがどのプログラムを指すのかすでに気が付いているかもしれません。そうです。

第1-1図のプログラム

がそれです。そういえば，我々はまだ**第1-1図**のプログラムを走らせていませんでしたね。

それでは，いよいよ実験開始です。

①**第1-1図**のプログラムを入力する。
②GD000↗

とキー・インする。

手順は以上の通りです。さあ，何が始まりましたか？何と

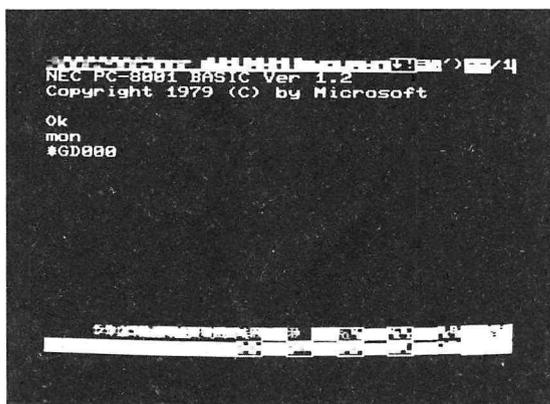
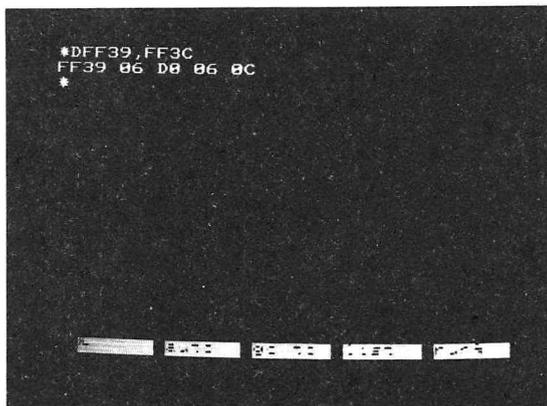
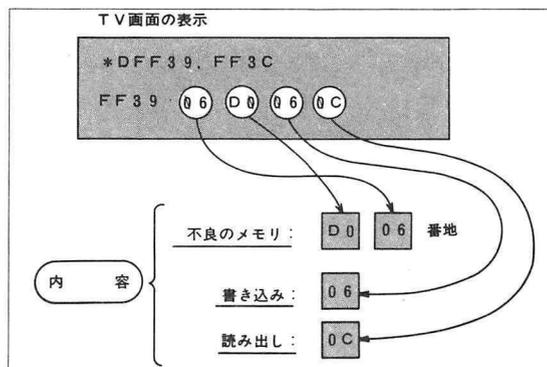
TMコマンドが実行

されているではありませんか(写真15)。プログラムがスタートし，しばらくすると画面が乱れ，やがて全てが消えます。TMコマンドと同じですね。さらにしばらくはお待ちください。約1分15秒後，

ピー

というブザー音が発生します。やった！

メモリ・エラーが発生

したのです(エラー発生を喜ぶなんて……)。

《写真15》TMコマンドが実行

《写真16》DFF39，FF3Cとキー・イン

《第1－19図》メモリ・エラーの見方

26. 不良箇所の診断

そこで，第2章では省略した

'メモリ・エラー発生'の対処法

を説明していくことにします。

①マシン本体の裏側にある**リセット・キー**を押す。（これでBEEP音が止まります）

②**MON**↗　でマシン語のコマンド・レベルに

③**DFF39，FF3C**↗　とキー・インする。

以上の操作で画面は，**写真16**のようになるでしょう。続いて**第1-19図**のようにTV画面に表示されたデータを並べ換えてください。今回の場合では，TMコマンド実行の結果，次のように解釈されます。

不良のメモリ：D006番地

そのメモリに記憶させたデータ：06

そのメモリから読んだデータ：0C

(注)　これはあくまでも，仮にエラーを発生させただけで，別にあなたのメモリが不良になったわけではありません。

さて，以上をさらに噛み砕いて言えば，

「D006番地のメモリに06というデータを記憶させたはずなのに，実際は0Cが記憶されていた。これはおかしい！」

というわけです。このようにTMコマンドにより，何番地のメモリが不良か確認できるのですが，実は不良の場所を更に細く分析することができるのです。それにはもう少し知識が必要なのですが，せっかくですからここで知っておくことにしましょう。

27. 8ビット並列処理

そのために，ここでいくつかの言葉の意味を理解してください。

> **CPU＝中央処理装置(Central Processing Unit)**
> コンピュータの中枢となる中央演算処理装置のこと。

ＣＰＵについては，有名ですから知っていますね。我々のマイコンの“頭”に相当する部分で，マイコンの場合は

たった一つの部品

からできています。これを世に

1チップＣＰＵ

と呼んでいます。

もちろん我々のマシンにもＣＰＵが使われています。現在，マイコンのＣＰＵは

80系（8080系）
68系（6800系）

という二つの系統に分れています。それは，

80系：8080ＣＰＵ（インテル）
68系：6800ＣＰＵ（モトローラ）

というマイコン初期における二大ＣＰＵの流れを組むもので，我々のマシンのＣＰＵは，

Z-80（80系）

というＣＰＵです。

さて，そのＺ-80というＣＰＵは

8ビット並列処理

のＣＰＵです。難しい言葉ですね。とにかく

8つの情報を同時に処理できるＣＰＵ

ということです。そして，前節まで見てきたメモリについても，

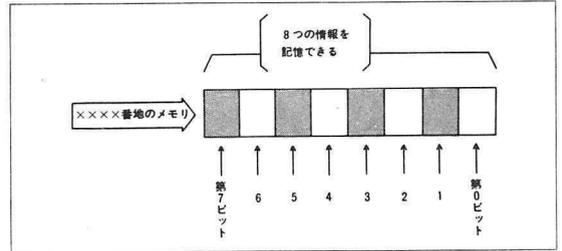
1つ1つのメモリが
8つの情報を記憶できる

ようになっています。その1つ1つの記憶場所には，

ビット

という名前がつけられていて，

第0ビット～第7ビット

で区別しています(**第1-20図**)。

《第1－20図》メモリとビッド

28. メモリ診断結果

以上で言葉の予備知識ができましたから，再びＴＭコマンドのエラー処理に戻ります。

第1-21図をみてください。左側には16進数が並んでいます。また右側には，０と１の数字が並んでいます。読者の大部分は，これが

16進数と２進数の対応表

であることはご存知だと思います。**2進数**については，コンピュータの本を読めばたいてい出てきますから，‘ああまたか’と**飽き飽き**していることだと思います。

0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
A	1 0 1 0
B	1 0 1 1
C	1 1 0 0
D	1 1 0 1
E	1 1 1 0
F	1 1 1 1

《第1－21図》16進数と1，0の対応

したがって，本書では**2進数については常識**とみなし，説明致しません。

そこで，この表を見ながら先のエラー・チェックで得られたデータを

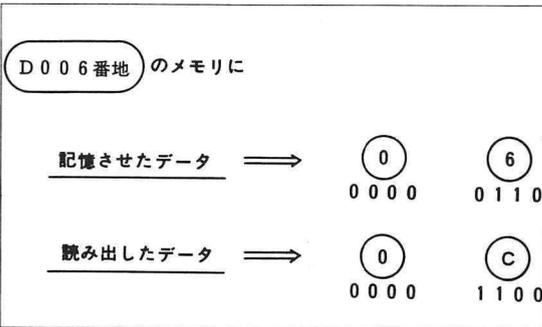
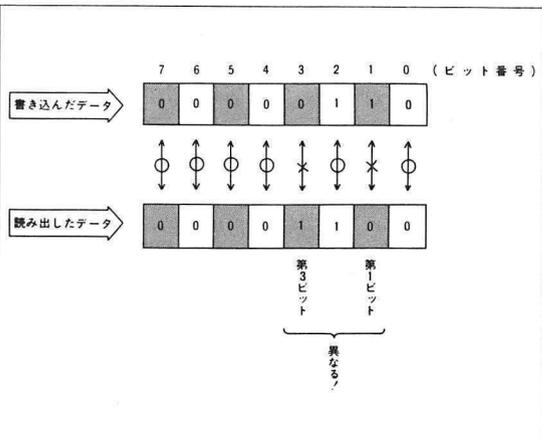
　　2進数に変換

します(**第1-22図**)。すると

　　0000　0110――06
　　0000　1100――0C

になります。こうして得られたデータを，ビット毎に比較してみます(**第1-23図**)。どうやら8ケ所のうち**2ケ所がおかしい**らしいことがわかります。さらに正確に言えば，

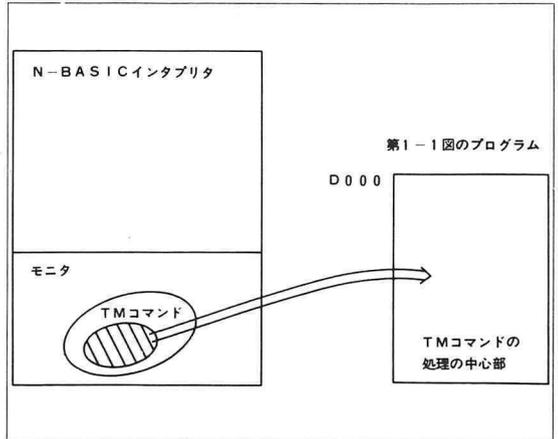
> **D006番地のメモリの**
> **第1・3ビットが不良！**

であったというわけです。

《第1－22図》TMコマンドで得られたデータを2進数に変換

《第1－23図》2進数でチェックする

さあ，これでTMコマンドの使い方については，すべて理解していただけたと思います。これで本章，および本ブロックの幕をとじても良いわけです。しかし，追求熱心なあなたはそれを許してくれないでしょう。そんなあなたに，次節を用意致しました。

29. メモリ・エラーの原因追求

本ブロックの最後に，**TMコマンド・エラー**を発生させた**第1-1図**のプログラムについてその仕組みを簡単に説明しておきましょう。

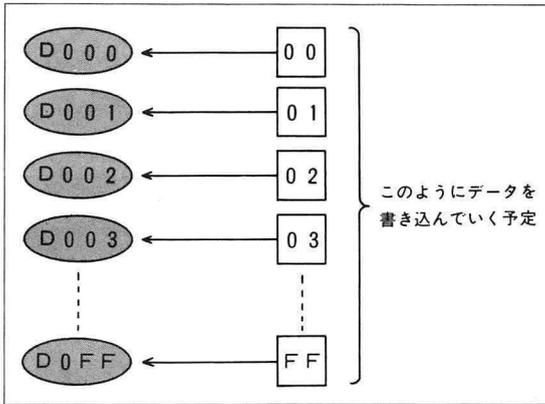
第1-24図のように，**第1-1図**のプログラムはモニタ内にあるTMコマンドの中から，**処理の中心**となる部分を取り出し，D000番地から書いたものです。このプログラムを走らせると，どんなことが起こるでしょうか？

《第1－24図》TMコマンド処理部を移す

① 第1章で説明しましたように，モニタはメモリのおしまいの方から256ずつチェックをしていきます。

② チェックがだんだん前の方に進み，やがて
　　D000～D0FF番地
をチェックする番になります。

③ チェックの要領は，**第1-25図**のようにまず各メモリに
　　00～FF
までの16進を書き込み(記憶させ)，次にもう1度その記憶した内容を読み出し，
　　00～FF
と合っているか調べるというものでした(前述)。

④ そして実際にD000番地から順に
　　00，01，02，……
とデータが書き込まれて行きます。

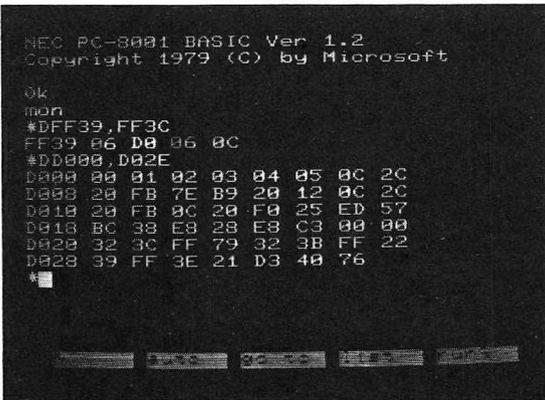
⑤ ところが，ところが――。本当は**D000番地には第1-1図のプログラムが入っていました。このため第1-1図のプログラムは，前の方から書き換えられていきます。**

《第1－25図》テスト・データを書き込む

⑥ 実際は，

　　００，０１，０２，０３，０４，０５

まで書き込み，Ｄ００６番地まで進んだ時，面白いことが起こります。それでは，**第1-1図**のプログラムを走らせ，エラーが起こったあとの様子を見てみましょう。

　　ＤＤ０００，Ｄ０２Ｅ↗

とキー・インしてください。確かにＤ０００番地からプログラムが書き換えられています。しかし，何とＤ００６番地からは，**第1-1図**のプログラムがそっくり残っているではありませんか(**写真17**)？。

《写真17》第１－１図のプログラムが残っている

⑦ その秘密は，Ｄ００５番地にあります。実は，

D005番地：メモリにテスト・データを書き込む命令

が入っていたのです。そして自分自身のプログラムのため

　　Ｄ００５番地＝05
　　　　　　　　↑
　　　　　　減算命令

に書き換えられてしまったのです。この減算命令では，このプログラムに関しては，まったく影響を与えません。

⑧ このため，Ｄ００６番地以降については

　　０６，０７，……，ＦＦ

のテスト・データは用意されるのですが，**メモリにはまったく書き込まれない**のです。

⑨ テストの方は，そんなことにはおかまいなしに，読み出しチェックに入ります。そして，

　　Ｄ０００番地＝００？
　　Ｄ００１番地＝０１？
　　Ｄ００２番地＝０２？

と比較していきます。

⑩ そして，――。もうおわかりですね。Ｄ００６番地に達したとき，

　　記憶させたはずのデータ＝０６｝合わない
　　読み出してみたデータ＝０Ｃ｝

このため，エラーが発生したわけです。

ここまでで，第１ブロックの全てを終了します。読者の方は，本書の第１ステップを見事読破したことになります。

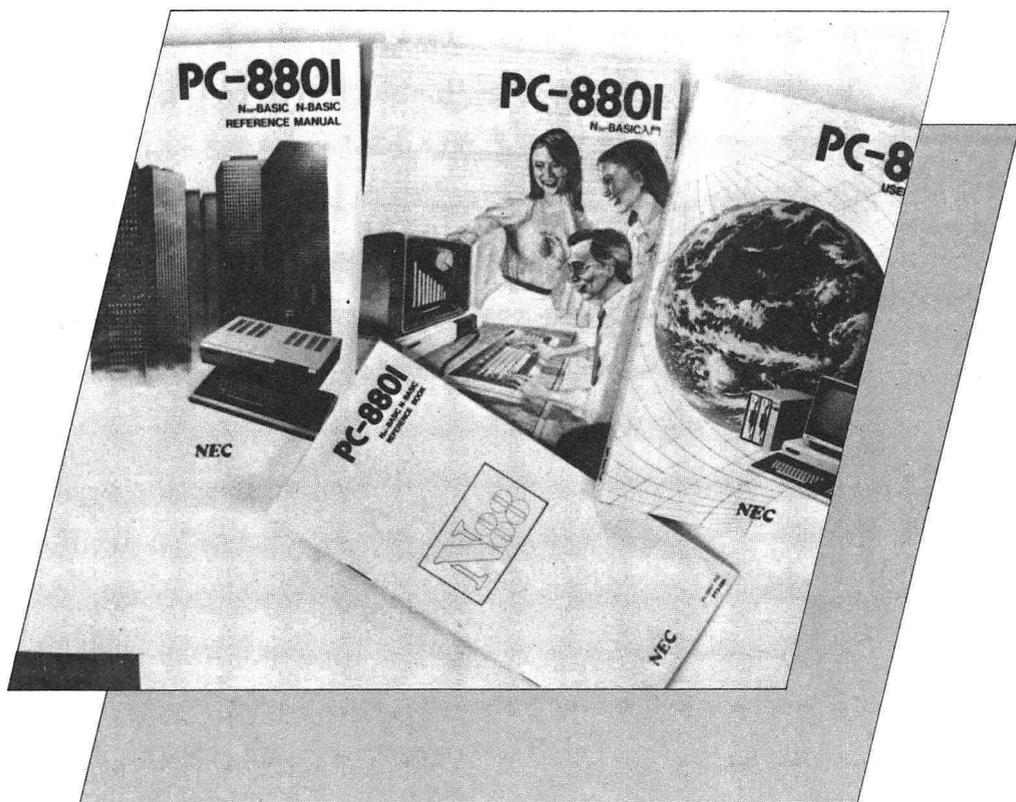
〈第４章のおわりに〉

さあ，やりましたね。ついに来ましたね。しかしここで挫折しないでくださいね。なにしろ次ブロックから，マシン語の命令そのものに入っていくのですから。ジャン，ジャン。

年	月	内容	
80	11	トレード・ゲーム	
81	7	インディアン・ポーカー	※
	8	PC−8001 マシン語セミナ開催記	
		エイリアン・ビリヤード	※
	10	ピラミッドとミイラ	※
	11	N−BASICの内部構造・中間言語を探る	
	12	スーパー・ムービング・ブロック	※
82	1	（別冊付録）Z80 マシン語活用マニュアル	
	2	松下JR−100の性能を探る	
	3	シンクレアZX−81の紹介	
GAMINGへの招待（連載） シリーズ1　81年12月～82年4月 シリーズ2　82年5月～			
マイコン・ソフト・パッケージ（PC−8001） クリンゴン・キャプチャー・パートII マルチ・ユース・パーソナル・データベース （付属のカセットに入っています）			

（※）は、カセット・サービスで入手可能です。

第2ブロック

マシン語プログラミングの実践1

ま・し・ん・ご・ノ・ぷ・ろ・ぐ・ら・み・ん・ぐ——デキルカナ?

第1章

マシン語のプログラムに初挑戦

〈はじめに〉

いよいよ第2ブロックのスタートです。本章において，**一つの完全なマシン語が呈示さ**れます。それを理解することが，本章の目的となります。覚悟のほどは，いかが？

1. 我々のわかること

本節から，いよいよ

マシン語のプログラム

に挑戦していくことになります。いわば本節が我々の

“マシン語の旅”

への出発点となるわけです。それがあなたにとっての

楽しい「マイコン」の旅

になるか，

いばらと苦痛の旅

になるか，現時点ではまったく予測がつきません。しかし，それが困難の旅であればあるほど，目的地に着いた時の喜びは大きいものになるでしょう。本書を読まれる全ての皆さんが，途中で挫折することなく，しかもできれば楽しい旅を送れるよう期待しています。それではいよいよ，その

マシン語への旅

へ出発することに致しましょう。

第2－1図をみてください。

番地	マシン語	番地	マシン語
D000	3E	03	00
01	11	04	E0
02	32	05	76

《第2-1図》マシン語のプログラム（既知の形）

すでに我々はたくさんの予備知識を持っています。たとえばこの図の中にある

番　地

とか

マシン語

とかの意味はわかります。またこの図のプログラムをあなたのマシンに入力することもできるし，そのプログラムを走らせることもできます。さらにそのプログラムをカセットに録音することもできるし，それを読み込むこともできるわけです。

つまり我々は，**すでに多くのマシン語についての予備知識を持っている**ことになります。しかし，あなたにはまだその実感がわかないかもしれません。

なぜでしょう？

おそらく次の理由によるのではないでしょうか？

すなわち，我々は**第2－1図**のプログラムについて，いろいろな事ができる。さまざまに**扱う**ことはできる。しかし，しかし——。我々は**第2－1図**について，

その中身はまったくわからない！

のです。つまり，我々はマシン語の

プログラムの扱い方

はわかるが，

プログラムそのもの

については何一つ知らないのです。マシン語のプログラムの組み方も知らないし，解析のやり方もわからない。結局，我々は**第2－1図**のプログラムについては，何もわかっていなかったのです。

2. マシン語の二つの形態

結局，これから我々がやろうとしている目標は，

第2－1図のプログラムの内容が

わかるようになること

ということになります。

そこで，——。

これから我々は，直ちに

マシン語のプログラムを組んでみる

ことになります。**紙とエンピツ**のご用意をお願いします。OK？

それでは，**第2-2図**をみてください。

アセンブリ言語
LD　A, 11H LD　(0E000H), A HALT

《第2-2図》マシン語のプログラム（未知の形）

何だか**チンプン・カンプンな事**が書いてありますね？結構。直ちにこれを，あなたの用意した紙の上に写してみてください。

いかがですか？

今あなたが写したその

チンプン・カンプンなもの

が，**マシン語のプログラム**です。

と言ったら驚くでしょうか？　なにしろ我々は第1ブロックにおいて，いろいろマシン語をいじくってきました。そのマシン語は，**第2-1図**のような形をしていたはずです。それは**第2-2図**とは，まるで違った形をしています。それを，**第2-2図**がマシン語のプログラムだと言われても今さら——。

しかし，**第2-1図**も，**第2-2図**も，

どちらもれっきとしたマシン語のプログラム

です。すなわち，**マシン語には二つの形がある**のです。

3. どちらを選ぶ？

まだピンとこないかもしれません。しかし，マシン語には二つの形態があって，

一つは，**第2-1図**のような形

をしているし，

一つは，**第2-2図**のような形

をしているのです。そして，どちらもマシン語のプログラムです。しかも，今述べたように

どちらもまったく同じ内容の

マシン語のプログラム

なのです。信じられますか？　しかしこれは事実です。

そこで感覚テストです。あなたにとっては，**第2-1図**のマシン語のプログラムと**第2-2図**のマシン語のプログラムを比べると，どちらの形が**わかりやすいですか？**　どちらの方が，**感覚的にピンときますか？**　どちらの方が**覚えやすいですか？**

答は人によってマチマチでしょう。もしくは，まだピンとこないかもしれません。

第2-1図の形は，数字（16進数）の羅列です。それに対して，**第2-2図**のプログラムでは何やら**英語の単語のようなもの**が並んでおり，BASICとは言わないまでも，どちらかといえばいかにも**プログラムらしい形**をしています。

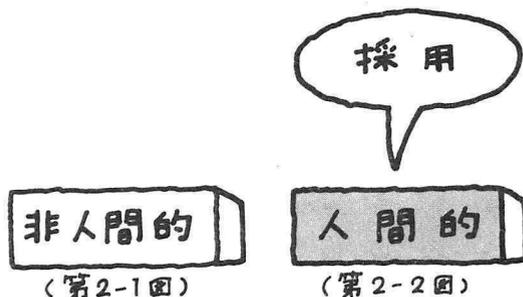
第2-1図にしても，**第2-2図**にしてもどちらも非常に短いプログラムです。したがってどちらを覚えようとしても大差ないかもしれません。しかし，やがてもっと大きな，膨大なプログラムを相手にするようになった時，おそらく我々人間は，**第2-1図**のような数字の羅列ではまいってしまうでしょう。もちろん，暗記するにも困難をきたすことでしょう。

我々は，

マシン語には二つの形態がある

ことを知りました。そして，これからどちらかの形でマシン語を学んでいくわけです。どうせ学ぶなら，楽な方が良い。そこで，我々は

後　者

すなわち**第2-2図**のような形態を採用し，その書き方でマシン語を学んでいくことにします。

4. 二つの基本概念

そこで，これから**第2-2図**のプログラムについて学んでいくことにします。

まず，1行目をみてください。真中のところに

　　A

の文字が見えます。これを**レジスタ**といいます。

レジスタ(register) 命令，データなどの情報を，一時たくわえておくための**装置**

いかめしく書くと上記のようになりますが，ここではとりあえず

　　レジスタとはBASICの**変数**のようなもの

と覚えてください。正確には，BASICの変数はメモリ上にあり，レジスタはCPUの中にある装置というハード上の違いがありますが，ここでは気にしないでおくことにします。

後でまとめますが，レジスタにはいろいろな種類があって，

　　××レジスタ

と呼んでいます。たとえば今のAでしたら，

　　Aレジスタ

というわけです。

次に〝**バイト**〟という言葉を覚えてください。

前ブロックにおいて，我々は，

　　0 1 2 3 4 5 6 7 8 9 A B C D E F

の16の英数字で構成されている

　　16進数

という言葉を覚えました。

さて，その

　　16進数2桁のことを1バイト

と言います。たとえば，

　　1 2

　　1 A

　　0 9

　　F F

はそれぞれ皆1バイトの数です。もし16進数4桁になれば，2バイトの数になります。たとえば，

　　1 2 3 4

　　A B C D

はどちらも2バイトの数だし，

　　0 0 A B 9 4

は3バイトの数です。

バイト(byte) 情報量の大きさを表わす単位。普通16進数2桁の情報量を1バイトと呼ぶ。

以上，本節ではマシン語を学ぶ上で基本になる

　　レジスタ

　　バイト

の二つの言葉を覚えました。次節では，この二つの概念をつなげてみることにします。

5. Aレジスタは8ビット・レジスタ

そこで，**第2-2図**を見ていくと，

　　L D

という命令が2回出てくるのがわかります。

LD(ロード命令) **〈書式〉**　L D　X_1，X_2 **〈機能〉**　$X_1 \leftarrow X_2$

LD命令はBASICの

　　L E T

に似ています。そこで，

　　L D　X_1，X_2

は，BASICの

　　L E T　$X_1 = X_2$

と同じ機能を持つとお考えになって結構です。すなわち左辺のX_1に右辺のX_2が代入されるわけです。

以上を理解した上で**第2-2図**の1行目を見てください。今のLD命令の説明によれば，この行の意味は，

　　Aレジスタに11Hが代入される

ということになるでしょう。この場合疑問になるのは，右辺の

　　1 1 H

です。〝11H〟というのは一体何でしょう。

〈習　慣〉

コンピュータの世界では，**16進数のあとにはHをつける。**

普通，10進数と16進数を区別するため，

16進数のあとには，Hをつける。

きまりになっています。

さあ，これで１１Hの意味がわかりました。

１１H＝16進数の１１

ですね。すると**第2-2図**の１行目の意味は，

Aレジスタに16進数の11を代入する

ということになります。

ここで重要な注意があります。

先に我々は，レジスタは変数のようなものということを知りました。しかし，いくら変数のようなものといってもそこはBASICと違い，やたらな数を代入することはできません。たとえば，

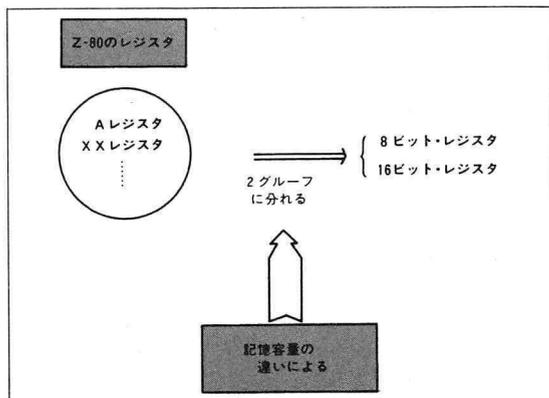
ＬＤ　Ａ，１２３４Ｈ

のようなことはできません。

我々のマシンのＣＰＵには，Ｚ－80が使われているというのは覚えていますね。そのＺ－80は，内部にいくつかのレジスタを持っています。そしてそれらのレジスタは，その記憶容量から二つのグループに分けることができます。

Ｚ－80のレジスタ

８ビット・レジスタ
１６ビット・レジスタ

の二グループです（**第2-3図**）。

《第2-3図》8ビット・16ビット・レジスタ

ここで注意しなければならないのは，

8ビット・レジスタ＝8ビットの数を記憶

16ビット・レジスタ＝16ビットの数を記憶

するということです。

順に説明致します。

ビット（bit）

記憶装置の情報容量の単位で，２進数の１桁のこと。８ビット＝１バイト

ビットというのは，binary digitの略で，

8ビット＝1バイト

という換算式を覚えておいてください。次もわかりますね？

8ビット＝16進数2桁の情報量

16進数1桁は4ビットで表わせる

これらの換算は全て重要です。

ビットがわかったところで，先の２種類のレジスタに戻ってください。今のことから次のことがわかると思います。

８ビット・レジスタ＝**16進数2桁**を記憶

１６ビット・レジスタ＝**16進数4桁**を記憶

ところで

Aレジスタは8ビット・レジスタ

です。したがって次のことがわかると思います。

ＬＤ　Ａ，１２Ｈ――――（可能）

ＬＤ　Ａ，１２３４Ｈ―――（不可能）

(注)　　ＬＤ　Ａ，１Ｈ

というのは可能です。これを見るとあたかもＡレジスタに**4ビット**の情報が記憶されたように見えます。しかし，**Aレジスタは8ビット・レジスタ**ですから，実際は上位に０が入り

ＬＤ　Ａ，01Ｈ

のように代入されます。

6. （０Ｅ０００Ｈ）とは？

次に第２－２図の２行目を調べてみます。

２行目もＬＤ命令です。意味は，

〝（０Ｅ０００Ｈ）にＡを代入する〞

ですが，この中の（０Ｅ０００Ｈ）がわからないと思います。説明致しましょう。

①Ｈは16進数のこと

（　）の中のＨは，すでに見てきたように16進数を表わしています。したがって，０Ｅ０００Ｈは16進数です。

②Ｅ０００は番地のこと。

ＬＤ命令の左辺に２バイト（16ビット）の16進が出てきたら，それは番地を表わします。ですから，Ｅ０００番地というわけです。

③（　）のある・なしでは意味が異なる

今後，マシン語を学んでいくにしたがって，２バイトの16進数は，

０Ｅ０００Ｈ

（０Ｅ０００Ｈ）

の二つの形が出てきます。もちろん（　）のある・なしでは意味が異なります。その違いを大まかに言えば，

０Ｅ０００Ｈは，Ｅ０００そのものを指し

（０Ｅ０００Ｈ）は，Ｅ０００番地の中身

を指しています。これだけではピンときませんね？ところで，ＬＤ命令の左辺には（　）のついた形しか現われません。そこで（　）のある・なしについては，ＬＤ命令の右辺に２バイトの数が現われたとき説明することにします。

④数の先頭は常に数字

何だか当り前のことを言っていますね。この概念は重要ですから，良く読んでください。

16進数は，

０１２３４５６７８９ＡＢＣＤＥＦ

の16の**英字**と**数字**を使います。ところで

Ｅ０００Ｈ

の先頭は数字ですか？　いいえ，英字ですね。このように16進数を書くときは，

数の先頭に**英字**がくることがある

わけです。こんなときは，先頭に０をつけ

数の先頭は常に数字

になるようにしています。

もっとも，あなたがメモ用紙にプログラムを書くときはいちいちそこまで注意をしなくても結構です。しかし，先頭に英字がきたときは０をつけるのが正式な書き方です。また，将来あなたが**「アセンブラ」**というプログラムを使う日がくれば，正式な書き方をしないと**エラー扱い**にされます。

以上，本節は２行目のプログラムを解析してみました。まだ納得できない点があるかもしれませんが，あとでもう１度まとめますから，もう少し我慢して読み続けてください。

7. HALTは「ハルト」ではない！

第２－２図の最後，３行目のプログラムです。

HALT 命令 ＣＰＵをＨＡＬＴ状態にする。

ＨＡＬＴは，英語で「停止する」という意味ですね。すなわちＨＡＬＴ命令は，

BASIC のEND

に相当する命令で，プログラムの終りや途中で止めたい場所に使います。

なお，ちなみにＨＡＬＴを「ハルト」と読む人が時々いますが，「ホールト」の方が原音に近いようです。参考までに発音記号を示すと，

hɔ́:lt

となります。

8. まとめると

以上で**第２－２図**の個々の命令はわかりましたので，全体の意味を考えてみましょう。

①１行目

Ａレジスタの値が１１Ｈになります。

②２行目

Ｅ０００番地の内容が，Ａレジスタの値，すなわち１１Ｈになります。

③３行目

プログラム終了です。

第2-2図のプログラムは，変数であるAレジスタに１１Hを代入し，本当に代入されたかを確認するためにAレジスタの中身をＥ０００番地に入れるというものです。したがって，このプログラムを走らせたあと**Ｅ０００番地の値を見て１１Hになっていれば，プログラムがうまく動いた**ことになります。

さあ，**第2-2図**のプログラムの意味はお分りになったと思います。そこでこのプログラムを実際にマシンにかけて走らせてみたいのですが，どのように入力したら良いでしょうか？

我々は，前ブロックにおいて**第2-1図**のようなプログラムを入力する方法を知りました。しかし，**第2-2図**のようなプログラムについては，まだ未知です。

——結論から先に申し上げましょう。

第2-2図の形のプログラムは，マシンに入力することはできません。そこで，我々は何らかの形で

第2-2図のプログラムを

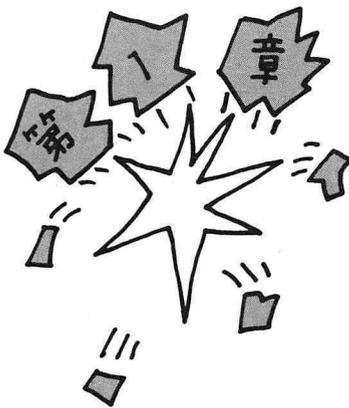
第2-1図の形のプログラムに変換

する必要があります。次章は，その変換の仕方から入っていくことに致します。

〈第1章のおわりに〉

本章において，とにもかくにも我々は，

一つの完全なマシン語のプログラムの

解析に成功しました。あとは，それを第1ブロックで学んだ知識とドッキングさせるだけでそのプログラムを走らせることができます。それが第2章の目的となるわけです。

第2章 ハンド・アセンブルの実践

<はじめに>

私は，その日の気分によっていろいろな言語でプログラミングを楽しんでいます。それは高級言語からマシン語まで，それこそ気まぐれで何でもかじります。まあ，90％マシン語を使うことが多いのですが，Ｚ－80だけでなく，68系や，16ビット，果ては大型機のアセンブラまで楽しんでいます。

もちろん，それら全部が自分のマシンで走るわけではありません。むしろ走る方が少ないのは当然です。そして，走るものは実際にマシンにかけて結果を見るし，走らないものはただ**紙上プログラミング**で我慢しているわけです。

こうしていろいろなＣＰＵのマシン語，あるいは高級言語をいじくっていると，それぞれのＣＰＵの特徴もわかるし，また高級言語の長所・短所，向き・不向きもわかってきます。

さて，こうしていろいろな言語，マシン語を日を変えては接しているうちに，二つのことに気づきました。

［この続きは，本章の最後に書くことにします。そうしないと，いつまでたっても第２章が始まりませんので。］

9. アセンブラとハンド・アセンブル

アセンブリ言語(assembly language)

プログラミング言語のうち，もっとも機械語に近い言語。**機械語の命令と１対１対応**している。

〝**アセンブリ言語**〟――この言葉をどこかで見たことはありませんか？　実はあるのです。**第２－２図**を良く見てください。表頭に〝アセンブリ言語〟と書いてありますね。

アセンブリ言語は，**プログラミング言語の一種**ですが，

機械語そのものではない

のです。しかし，ここが重要なところですが，

機械語の命令と**１対１対応**している

ため，

アセンブリ言語によるプログラム
＝マシン語のプログラム

と考えても不都合は生じません。アセンブリ言語とマシン語を比べた場合，第１章で見たように**アセンブリ言語の方が人間にとってはわかりやすい**ですね。そこで通常，マシン語のプログラムを作るときは，まずわかりやすい**アセンブリ言語で組みます**。しかしそのままではマシンに入力することはできませんから，一度**マシン語に変換するという作業**を行い，それから入力，実行という手順になります。これを図示すると**第２－４図**のようになります。

《第２-４図》マシン語のプログラムを作る

アセンブリ言語からマシン語に変換する作業を，

アセンブル作業

と呼んでいます。その方法は，

手作業で行う＝ハンド・アセンブル

アセンブラを使う

の2通りがあります。

> **アセンブラ(assembler)**
> アセンブリ言語をマシン語に変換する言語翻訳プログラム。

本書では，アセンブル作業をもっぱら泥臭い**手作業**(**ハンド・アセンブル**)で行います。アセンブラに比べ多少は面倒ですが，ハンド・アセンブルを繰り返すことによりマシン語のしくみがわかってきますし，学び初めの人はやはり横着せず，基本から努力すべきでしょう。

10. Z－80活用表を用いて

そこでこれからハンド・アセンブルを行っていくことになります。それには道具が必要です。といっても，本書の付録1「**Z－80活用表**」があれば十分ですから，それをいつでも見られる状態にしておくと便利でしょう。

それでは**第2-2図**のプログラムを1行ずつハンド・アセンブルしていきます。まず**LD 命令**ですね。「Z－80活用表」の**8ビット**のところをみてください。左の列に命令が並んでいます。その1番上のところに

LD　A，X

というのがありますね。Xについては表頭にいろいろなものが並んでいます。このうち，***n***というところを見てください。

3E・***n***

と出ています。ここで***n***は，1バイトの数を表わしています。したがって**第2-2図**の1行目は，

LD　A，11H⟶3E　11

と変換されるのがわかります。

続いて2行目に進みましょう。「Z－80活用表」では，

n――1バイトの数

n n――2バイトの数

を表わしています。0E000Hは，2バイトの数ですから表の中の

LD　(***nn***)　と　A

の交わるところを見ると，

32・***n n***

とあります。したがって2行目は，

LD　(0E000H)

⟶32　E0　00

と変換されます――。

否，ダメダメです！

ここで重大な注意があります。

11. 80系特有の注意

前ブロックにおいて，我々のマシンのＣＰＵは，

Ｚ－80

であることを知りました。Ｚ－80は，

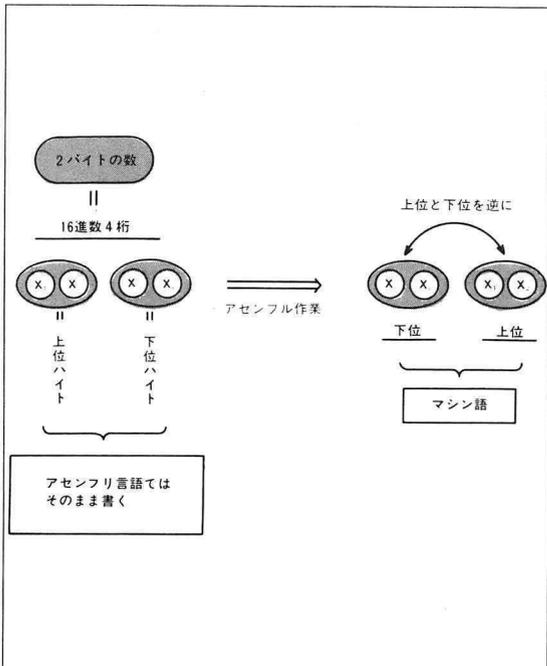
80系のCPU

です。そして，これから述べる注意は80系特有の注意です。

> **〈鉄　則〉**
> 80系のＣＰＵにおいては，２バイトの数をアセンブルする時，上位と下位を逆にする。

具体的に説明致します。

```
LD　(0E000H)
     2バイトの数
        ──→32　E0　00──×
               上位　下位
     (逆にする)
        ──→32　00　E0──○
```

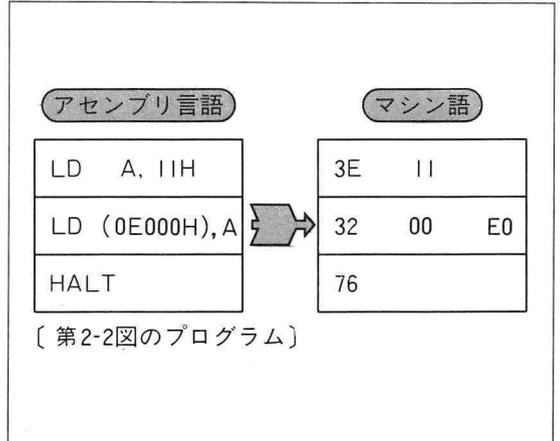
これが正しいアセンブル作業です(**第２－５図**)。

《第２－５図》80系特有の注意

これから我々がハンド・アセンブルするときは，この注意をゆめゆめ忘れないようにしてください。

12. プログラムをメモリ上に割り当てる

ハンド・アセンブルの最後は，３行目，ＨＡＬＴです。これは，「**CPU コントロール**」の２行目にあります。

ＨＡＬＴ──→７６

以上で，すべての行のハンド・アセンブルが終りました(**第２－６図**)。

《第２－６図》アセンブリ言語→マシン語

次に我々がしなければならないのは，

プログラムをメモリ上に割り当てる

という作業です。

> **メモリ(memory)**
> コンピュータの主要構成部分の一つで，コンピュータの処理に必要なプログラムやデータを記録する装置

我々のマシンの中には，メモリという記憶装置が入っており，たくさんのデータやプログラムを記憶できるようになっています。これらのたくさんの記憶場所には，それぞれ**２バイト（16進数で４桁）の数を対応**させて区別しています。これが前ブロックで見た

番　地

です。

> **番　地＝アドレス(address)**
> コンピュータ・システムでメモリの位置を示すのに使う。

さて，メモリ上の個々の位置は，16進数４桁が対応しています。ところで

16進数１桁は16通りの値をとる

ことができますから，16進数４桁では

16×16×16×16＝65,536（通り）

の値を取ることが可能です。したがって我々のマシンは，**65,536個の記憶場所**を持っていることになるわけです。それらを番地で示せば，

００００番地
～
ＦＦＦＦ番地 ──── ①

になります。

さて，我々がハンド・アセンブルしたプログラムを，①上のどこかに記憶させる（＝入力する）わけです。これを，**「プログラムをメモリ上に割り当てる」**と言います。

さあ，そこでどこかに割り当てるわけですが，①の範囲ならむやみやたらに割り当てて良いというわけではありません。それには，

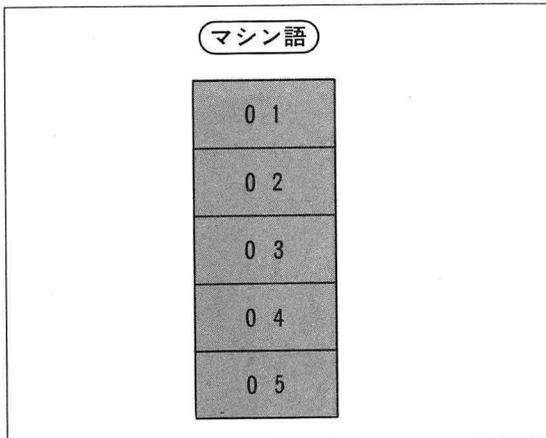
それなりの制限

があります。

13. 制限を見つける

これからその制限の意味を知るために，ある実験を行います。

第２－７図をみてください。これがその実験に使おうとするプログラムです。中身は，もちろんデタラメです。したがって，このプログラムを走らせても意味がありません。

さあ，そこでこれからこのプログラムをメモリのいろいろな番地に割り当ててみることにします。

《第２－７図》制限実験用プログラム

〈実験１〉

制限実験用プログラムを，Ｅ０００番地に割り当てる。

①**第２－８図**が，Ｅ０００番地に割り当てたものです。このプログラムを，Ｓコマンドであなたのマシンに入力してみてください。

②続いて，

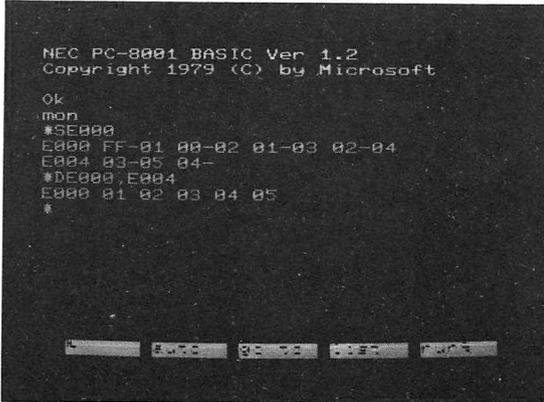
ＤＥ０００，Ｅ００４↗

で確認してみてください。

番　　地	マシン語
Ｅ０００	０１
〔注〕　０１	０２
０２	０３
０３	０４
０４	０５

(注)　これは、Ｅ001番地を省略して書いたもの。手書きのとき、よくこのように略すことが多い。今後は、この書き方を用いる。

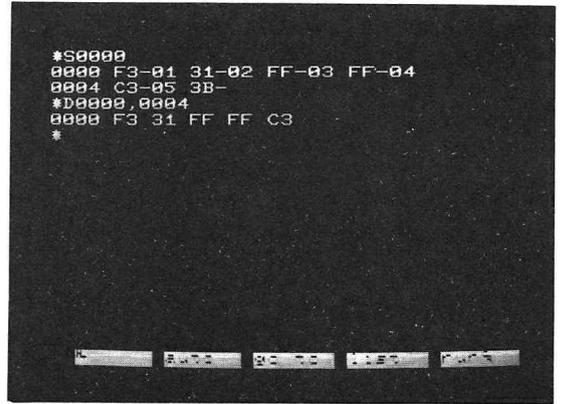
《第２－８図》Ｅ000番地に割り当てる

この実験については，特に問題はありませんね。**写真1**の通りです。正しく入力されていますね。

《写真1》正しく入力されたプログラム

〈実験2〉
第2-9図のように，制限実験用プログラムを0000番地から割り当てたものを使って同様の実験をする。

番　　地	マシン語
0000	01
01	02
02	03
03	04
04	05

《第2-9図》0000番地に割り当てる

今度は異変が起こります。まずSコマンドを使ってプログラムを入力する――ここまでは，問題はありません。次です。Dコマンドを取ってみる――なんと，**確かに書き込んだはずの第2-9図のプログラムが入っていないではありませんか！**　メモリは，プログラムを入力する前と同じ状態を保っているではありませんか**(写真2)**！

《写真2》プログラムが入力されない

これは，何度やってもダメです。プログラムは，入力されません。すなわち，

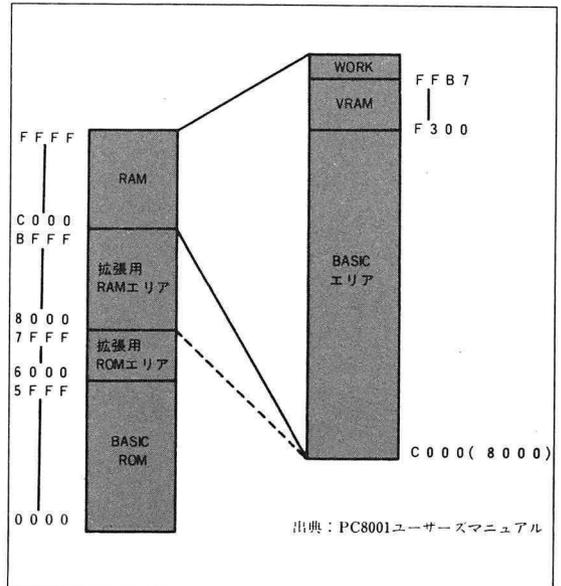
第2-7図のプログラムは，0000番地に割り当てることは不可能

という結論が得られたわけです。

14. N－BASICを破壊する

なぜ，

E000番地に割り当てる――可
0000番地に割り当てる――不可

なのでしょう？――理由があります。

《第2-10図》メモリアドレスマップ

第2-10図をみてください。これが，あなたのマシンの

全メモリ

の様子を表わしています。これと同じものが付録にも掲げてあります。図を見ると，

００００～５ＦＦＦ番地―――――①

が〝BASIC ROM〞になっています。すなわち①の部分に**N-BASIC**の本体が入っているのです。もし，この部分に別のプログラムを書き込んでしまったら，どうなるでしょう？　もちろんN-BASICは破壊されてしまいます。それは，困りますね。

そこで当然，あなたのマシンのこの部分にはプログラムやデータを書き込めないようになっています。そうです。これが，実験２でプログラムが書き込めなかった理由です。

第2-10図を良く見ると

ＲＯＭ

ＲＡＭ

という言葉が目につきます。

ROM (read only memory)
読み出し専用メモリ。通常の使用状態でデータの書き込みはできない。
RAM (random access memory)
すべてのアドレスに対し，等速のアクセスが可能で，書き込み・読み出しのできるメモリ。

メモリには，２種類あって

{ **ROM――読み出しのみ可能**
  RAM――書き込み，読み出し共に可能

であることは良くご存知だと思います。前者は，〝N-BASIC〞のように書き換えられては困るものに使います。

そこで改めて**第2-10図**を見ると，

＜ **メモリの前半（０００００～７ＦＦＦ）：ROM**
  メモリの後半（８０００～ＦＦＦＦ）：RAM

になっているのがわかります。したがって，まず

プログラムを割り当てるには

前半は不適当

というのが理解できます。

15. 記号との遭遇

前節により，マシン語のプログラムをメモリに割り当てるには，後半のＲＡＭの部分

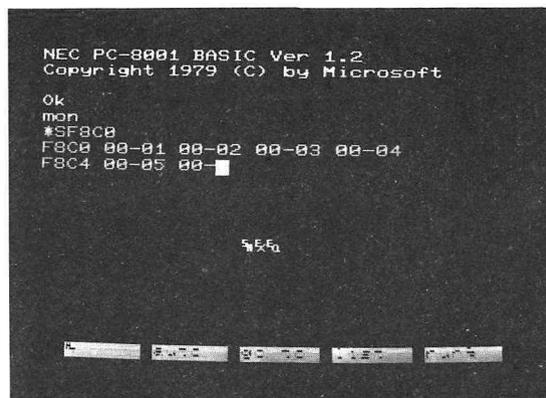
８０００～ＦＦＦＦ番地

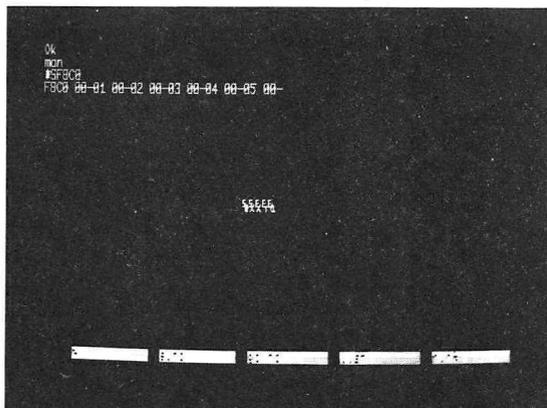
が良さそうなのがわかりました。しかし，まだこれだけの注意では不十分です。

〈実験３〉
① リセット・ボタンを押す。
② ＭＯＮ↗でマシン語のコマンド・レベルにする。
③ Ｆ８Ｃ０番地に割り当てた**第2-11図**プログラムを，Ｓコマンドで入力する。

番　　地	マシン語
Ｆ８Ｃ０	０１
Ｃ１	０２
Ｃ２	０３
Ｃ３	０４
Ｃ４	０５

《第2-11図》Ｆ８Ｃ０番地に割り当てる

おそらくこの実験をするとビックリするでしょう。**写真３**がその様子を表わしています。プログラムを入力するにしたがって，変な記号が画面の中央に現われてきます。

《写真３》変な記号が画面中央に現われる

今は，1行40字モードで実験しましたが，1行80字モードでも同じ実験をしてみてください。今度は，一つのマシン語を入力する度に，一つの記号が現われます(写真4)。

《写真4》 1行80字モードでの実験

我々の当初の予定では

F8C0番地はRAM上にある

のだから，この番地に割り当ててもよさそうな気がしました。しかし，その期待は実験3により見事に裏切られました。なにしろプログラムを入力する度に，画面の中央に記号が出現するのでは，無気味でたまったものではありません。

どうやら以上により，

RAM上にもプログラムを割り当てるのに不都合な場所があるらしい？

ことがわかってきました。

16. マシン語プログラム可能領域

さあ，実験はこれくらいにしてそろそろ結論を出しましょう。

今考えていることは，〝**マシン語のプログラムを割り当てるのに，どの番地が使えるか**〟という問題です。そして，その結論は次の通りです。

①0000～7FFF番地のROMの部分は，プログラムを書き込めないから不適当。

②EA00～FFFF番地は，システムのワーク・エリアとなっているため不適当。

③RAM領域の最初の32バイトは，割り込みサービスルーチンのジャンプ・テーブルとなっているため不適当

④以上を除いたエリアが，マシン語のプログラム・エリアに最適。

難しい表現になってしまいましたが，まとめると以上のようになります。順に説明しましょう。

①については，すでに実験で確認済みですからすぐにお分りになりますね。

> **ワーク・エリア＝作業領域**
> **＝ワーキング・ストレージ**
> **(working strage)**
> プログラムにおいて必要に応じてメモ的に使う記憶場所。

②のシステムとは，ここでは〝N－BASIC〟と〝マシンランゲージモニタ〟を指します。RAMエリアの終りの方は，これらのシステムが記憶場所に使っていますので，この部分（EA00～FFFF番地）は使わない方が安全です。

次に③の説明。**RAM領域の先頭は，第2-10図**を見てもわかるように通常は

8000番地

です。しかし，PC－8001の場合，標準装備ではRAMが16Kしかありませんから，まだRAMを増設していないときは，

C000番地

になります。そして，その最初の32バイト（16進数なら，00～1F）はやはり使わない方が安全ですから結論として

> **マシン語のプログラム領域**
> 16Kシステム：C020～E9FF
> 32Kシステム：8020～E9FF

となります。

17. ハンド・アセンブルのまとめ

さあ，これで〝**マシン語のプログラム可能領域**〟がわかりましたので，**第2-2図**のプログラムの**ハンド・アセンブルを完成**させましょう。

番地	マシン語	アセンブリ言語
D000	3E 11	LD A, 11H
02	32 00 E0	LD (0E000H), A
05	76	HALT

《第2-12図》ハンド・アセンブルの完了

プログラムは，プログラム可能領域でしたらどこでも良いのですが，ここでは16Kシステムの人でも，32Kシステムの人でも共に使える

D000番地

から割り当てることにします。**第2-12図**が，ハンド・アセンブルの完成したプログラムです。これと**第2-1図**を見比べてみてください。もはやこれらがまるっきり同じプログラムであることは，疑う余地がありませんね。**第2-1図は，第2-12図**のうち**アセンブリ言語の部分を省略したもの**だったのです。

以上をもちまして，ハンド・アセンブル作業は終了致しました。我々は，すぐにでも**第2-12図**のプログラムを入力し，走らせることができます。しかし，本章の目標は

ハンド・アセンブルの過程の修得

にありますから，ここでハンド・アセンブルの作業をまとめておきます。プログラムを走らせるのは，次章まで待ちましょう。

> **ハンド・アセンブル作業**
> ①アセンブリ言語を用いて，プログラムを作る（コーディングする）。
> ②〝**Z−80活用表**〟を見ながら，1行ずつマシン語におきかえていく。その時，2バイトのデータは上位と下位を逆にする。
> ③マシン語のプログラム作成可能領域に，今作成したマシン語のコードを割り当てる。

以上を完全に理解していただけたこととし，これで第2章を終わることに致します。

> **<第2章のおわりに>**
> （第2章のはじめから続く）
> それは，**自分のマシンで使える言語のマシン語はすぐ身につくが，本と紙だけで覚えた言語はすぐ忘れてしまう**ということ。もう1点は，ある言語をマスターしようとするとき**その言語を覚える必要はまったくなく，その言語のマニュアルの見方を知るだけで十分だ**ということです。
> 特に前者については，おそらくプログラミング言語の修得が，外国語の修得と似ているためでしょう。たとえば，**紙と本しかないの**と**話せる外人のいる**場合とでどちらが外国語を修得しやすいかは歴然としています。
> プログラミング言語についても同様で，マシンに不合理な入力をすると，マシンは受け付けてくれず，文法エラーがわかるわけです。いわば**マシンが，言語の教師みたいなもの**です。
> 以上のことから，私は実践を重視することにしています。また本書も基本的に実践を通して身につけていただく方針をとっていますので，あなたもサボらず，せっせとマシンに入力し，マシン語を走らせていってください。

第3章
レジスタを求めて

〈はじめに〉

私は前章において

実践を重視する！

と申し上げました。ところが，マシン語のプログラムで何かを実践しようとすると，やはりそれなりの予備知識が必要です。それは，次の二点です。

① **マシン語のオペレーション**

マシン語の実践をするには，やはりマシン語の扱い方を知らなくてはできません。

② **ハンド・アセンブル**

いくらマシン語のプログラムを作ってもそれをマシンに入力できる形に変換できなければ，①の知識が利用できません。

①については，すでに**第1ブロック**でマスターしています。また前章において我々は，

アセンブリ言語

マシン語

に変換するハンド・アセンブル作業をマスターしました。ということは，すでに我々は，

マシン語実践への準備は

完了した！

といえるわけです。あとは

実践あるのみ！

です。

というわけで，本章からは**新しい命令**が次から次へと出てきます。でも御安心ください。ムチャな出し方は致しません。かならず

〈チャレンジ〉——演習

を通して，具体的に理解していただくよう工夫しました。また本章の中で，興味のある人を対象に，我々の理解できる範囲内で

モニタの中身をのぞく

という試みもしています。どうぞ本章も楽しみに学習を進めていってください。

18. 処女航海

いよいよ我々の解析したプログラムを走らせることにします。

第2-12図のプログラムです。作業手続きはお分かりですね。Sコマンドで入力し，Dコマンドで確認してください(写真5)。ここでGコマンドで走らせるわけですが，その前に

チョット，待って！

ください。

《写真5》Sコマンドで入力，Dコマンドで確認

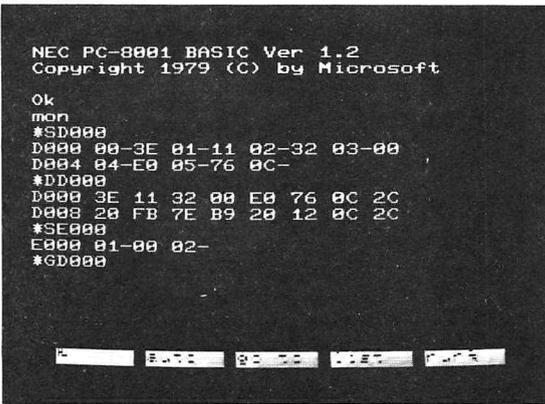
第2-12図のプログラムは，Aレジスタに11Hをセットし，それを確認のためにE000番地に入れるというものでした。したがって，プログラムを走らせたあとE000番地の値は11Hになるわけです。そこで確認のために，あらかじめE000番地に**他の値をセット**しておくことにします。ここでは，仮に00Hをセットすることにしましょう。

SE000↗

00↗

でセットしてください。これですべての準備が完了しました。いよいよ，走らせますよ。

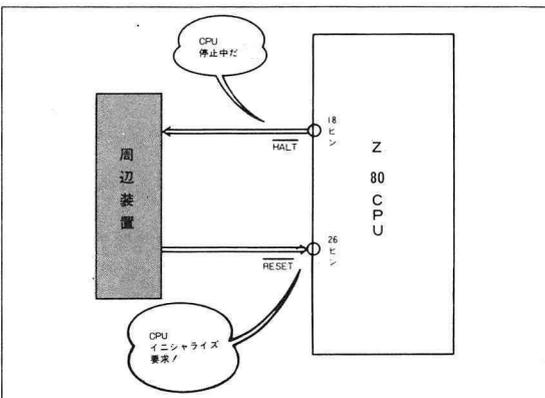
GD000↗

スタート！　何が起こりましたか？

《写真6》画面からカーソルが消えた

プログラムがうまく動けば，**写真6**のように

画面からカーソルが消え，

キーの入力を受け付けなくなる

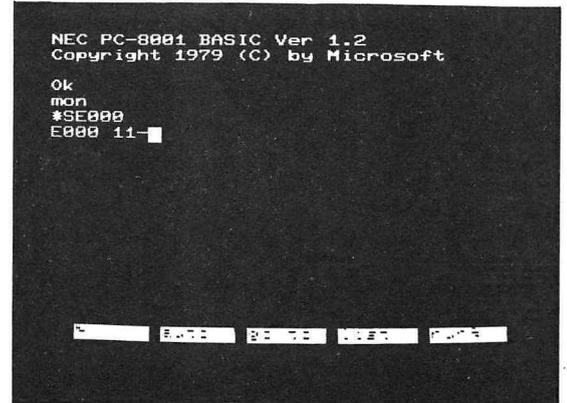
はずです。あなたのマシンは，死んだように動かなくなります。STOPキーを押してもダメです。これはどうしたことでしょう？　困りましたね。

《第2-13図》HALTとRESET

原因は，プログラム末の**HALT命令**のためです。この命令を実行すると，Z-80CPUは，18ピンに接続されているHALTの電圧をLOWに下げ，CPUが停止状態に入ったことを周辺に知らせます。この状態を解除するには26ピンにRESET信号を送ってやり，CPUをイニシャライズしてやります。

理屈ぽい話しを長々と述べましたが，早い話が，あなたのマシンの**リセット・ボタン**を押してください。これは，TMコマンドのところでもやりましたね。続いて　MON↗

でマシン語のコマンド・レベルにします。そして

SE000↗

としてください。**写真7**のように，11Hがセットされているのがわかります。すなわち，**第2-12図**のプログラムがうまく動いたわけです。

《写真7》11Hがセットされている

19. 全レジスタの登場

前節をもちまして，ついに我々は一つのマシン語の作成から始まって，ハンド・アセンブル作業，マシンへの入力，実行――と全ての工程を理解したわけです。マシン語を実践する上での必要知識は，全てものにしました。あとは

マシン語の命令をものにする

だけです(これが大変なのですが)。さあ，ドシドシ先に進みましょう。

マシン語の命令を理解するのに，変数に相当する

レジスタ

をどうしても避けて通ることはできません。そこで，第1章でみた**レジスタの全て**をここでまとめてみることにします。

種　類	記号	名　　　称	ビット
専用レジスタ	PC	プログラム・カウンタ	16
	SP	スタック・ポインタ	16
	IX	インデックス・レジスタ	16
	IY	インデックス・レジスタ	16
	I	インタラプト・ページ・アドレス・レジスタ	8
	R	メモリ・リフレッシュ・レジスタ	8
アキュームレータとフラグ	A	アキューム・レータ	8
	F	フラグ	8
	A′	アキューム・レータ(サブ)	8
	F′	フラグ(サブ)	8
汎用レジスタ	B	メイン・レジスタ	8
	C		8
	D		8
	E		8
	H		8
	L		8
	B′	サブ・レジスタ	8
	C′		8
	D′		8
	E′		8
	H′		8
	L′		8

《第2-14図》 レジスタの種類

第2-14図をみてください。これがZ－80レジスタの一覧です。全てマシン語の命令であやつることができます。これだけのレジスタをいきなり全て覚えることはできませんから，少しづつ使いながら覚えていくことに致しましょう。とりあえず，次のレジスタに焦点をしぼることにします。

アキューム・レータ：A
汎用レジスタ：B, C, D, E, H, L
専用レジスタ：IX, IY

そしてしばらく専門用語は不要です。〝×× **レジスタ**″と呼んでください。たとえば**IX レジスタ** のように。それともう一つ。そのレジスタが，

何ビットのレジスタ

であるか，常に注意を払ってください。なにしろ第1章で注意しましたように

各レジスタに代入できるビット数は
決められている！

からです。

20. 8ビット・レジスタ編

そこで，まず8ビットのレジスタから見てみましょう。次のプログラムに挑戦してみましょう。

〈チャレンジ1〉
Aレジスタ←―AAH
Bレジスタ←―BBH
Cレジスタ←―CCH
Dレジスタ←―DDH
Eレジスタ←―EEH
Hレジスタ←―F1H
Lレジスタ←―F2H
を代入するプログラムを作りなさい。プログラムは，D000番地から割り当てること。

さあ，チャレンジしてみましょう。あなた1人で。ハンド・アセンブルまでできれば，しめたものです。

作業としては，**まずアセンブル言語で記述する**のが良いでしょう。代入する部分は，LD命令を使えば良いですね。**プログラムの最後にHALTをつける**のを忘れないように。でき上がったアセンブリ言語によるプログラムが，**第2-15図**です。

アセンブリ言語
LD　　A, 0AAH
LD　　B, 0BBH
LD　　C, 0CCH
LD　　D, 0DDH
LD　　E, 0EEH
LD　　H, 0F1H
LD　　L, 0F2H
HALT

《第2-15図》 8ビット・レジスタを使う

次にハンド・アセンブルです。「Z－80活用表」を使うのでしたね。〝8ビット〟の〝LD〟のところまで，タテは各レジスタのところ，ヨコは〝8ビット〟の数を代入するのですから〝*n*〟のところを見てもらえば結構です。各命令のマシン語は次のようになります。

LD　A，*n*——3E
LD　B，*n*——06
LD　C，*n*——0E
LD　D，*n*——16
LD　E，*n*——1E
LD　H，*n*——26
LD　L，*n*——2E

またHALTは，既に見ましたようにマシン・コード＝76です。以上により，ハンド・アセンブルとしたものは，**第2-16図**のようになります。

マシン語	アセンブリ言語
3E　AA	LD　A, 0AAH
06　BB	LD　B, 0BBH
0E　CC	LD　C, 0CCH
16　DD	LD　D, 0DDH
1E　EE	LD　E, 0EEH
26　F1	LD　H, 0F1H
2E　F2	LD　L, 0F2H
76	HALT

《第2-16図》ハンド・アセンブル終了まで

最後にプログラムを，D000番地に割り当てます。完成したプログラムが，**第2-17図**です。これは，機械的に割り当てれば良いのですから，問題はないと思います。しかし，なかにはどうして2行目がD002番地になるのか，8行目がD00E番地になるのかわからない人がいるかもしれません。その人は，おそらく**16進数に慣れていない人**だと思います。なにしろ先にお断わりしたように，本書では16進数の説明を省略していますので無理はないと思います。16進数の表を片手に，マシン・コードを順に数えてみてください。

D000　3E
D001　AA
D002　06
⋮　⋮

番地	マシン語	アセンブリ言語
D000	3E　AA	LD　A, 0AAH
02	06　BB	LD　B, 0BBH
04	0E　CC	LD　C, 0CCH
06	16　DD	LD　D, 0DDH
08	1E　EE	LD　E, 0EEH
0A	26　F1	LD　H, 0F1H
0C	2E　F2	LD　L, 0F2H
0E	76	HALT

《第2-17図》8ビット、完成

なぜ2行目がD002番地になるのか，おわかりになりましたか？

これで〈チャレンジ1〉の解答を終ります。ここで**第2-17図**のプログラムを走らせるか否かは，あなたの自由です。しかし，本書ではもう少し先に進めてみることにします。

21. 16ビット・レジスタ編

8ビット・レジスタに続いて，本節では16ビット・レジスタを扱います。

> **〈チャレンジ2〉**
> ＩＸレジスタ←―１２３４Ｈ
> ＩＹレジスタ←―５６７８Ｈ
> を代入するプログラムを作りなさい。プログラムは，〈チャレンジ1〉のＨＡＬＴを取り，Ｄ００Ｅ番地から割り当ててください。

問題の意味は，わかりますね。〈チャレンジ1〉と〈チャレンジ2〉を１本のプログラムで作るのです。

できましたか？　**第2-18図**が一応完成したプログラムです。アセンブリ言語によるプログラミングは，特に問題はないでしょう。忘れやすいのは，

80系特有の注意点

です。２バイトのデータをマシン語に変換するとき，上位と下位を逆にすることに注意してください。

そこでハンド・アセンブル作業のところを説明しておきましょう。まず「Ｚ－80活用表」のひき方です。〝16ビット〟のＩＸ，ＩＹのＬＤ命令のところを見るのはわかりますね。ヨコについては，16ビットの数ですから〝*nn*〟のところをみます。結局，次のように変換されるでしょう。

ＬＤ　ＩＸ，*nn*――ＤＤ　２１
ＬＤ　ＩＹ，*nn*――ＦＤ　２１

それぞれ

２バイトのマシン語

に変換されることに注意してください。

次に，これにデータの部分をつなげます。上位と下位を逆にすればＯＫです。

ＬＤ　ＩＸ，１２３４Ｈ
――ＤＤ　２１　３４　１２
ＬＤ　ＩＹ，５６７８Ｈ
――ＦＤ　２１　７８　５６

以上のように，

８ビットと16ビット

では，若干マシン語の扱い方が異なります。**第2-19図**を参照してください。

番地	マシン語	アセンブリ言語
D000	3E AA	LD A, 0AAH
02	06 BB	LD B, 0BBH
04	0E CC	LD C, 0CCH
06	16 DD	LD D, 0DDH
08	1E EE	LD E, 0EEH
0A	26 F1	LD H, 0F1H
0C	2E F2	LD L, 0F2H
0E	DD 21 34 12	LD IX, 1234H
12	FD 21 78 56	LD IY, 5678H
16	76	HALT

《第2-18図》16ビット・レジスタを使う

22. 二つの課題

前2節におきまして，８ビット，16ビットの各レジスタにいろいろなデータをセットしました。そして，そのプログラムが**第2-18図**にまとまっています。お待たせしました。ここでこのプログラムを走らせてみることにします。

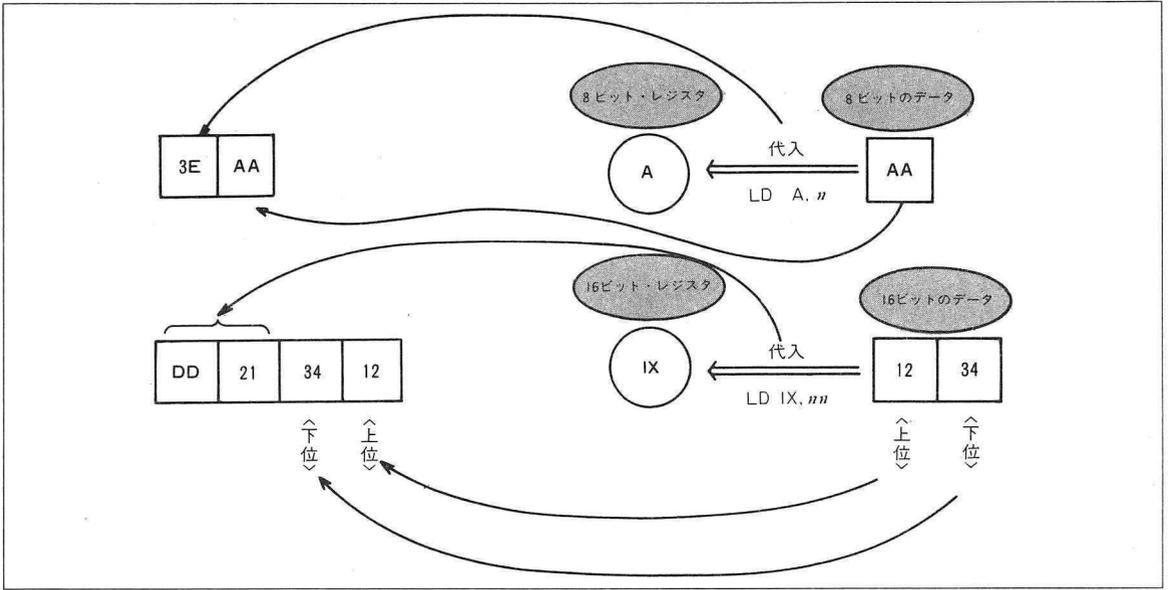
① Ｓコマンドでプログラムを，あなたのマシンにキー・インしてください。
② ＤＤ０００，Ｄ０１７↗
で，正しく入力されたか確認します。
③ 準備は整いました。プログラムを走らせます。
ＧＤ０００↗

さあ，何が起こりましたか(**写真8**)？　**第2-12図**の時と同じように，ＣＰＵはHALT状態に入り，カーソルが消え，キーの入力を受け付けなかったことと思います。しかし，もうあなたはあわてませんね。その対処法を知っているはずですから。

① **リセット・キー**を押す。
② **MON**↗　で〝*〟にする。

ＯＫですね。

さて，ここで二つのことに気がついたと思います。

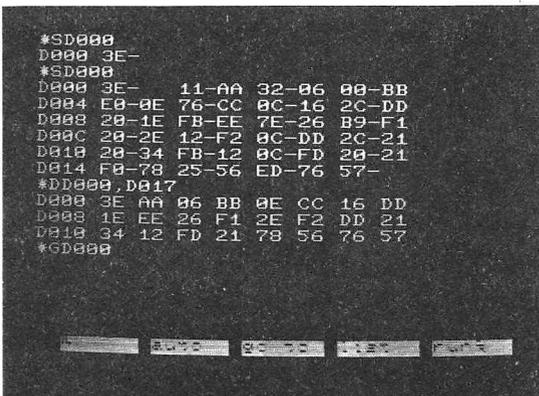
〈その1〉

《第２-19図》 ８ビット、16ビットの代入

マシン語のプログラムを走らせるたびに，いちいちうしろのリセット・キーを押さねばならないのか？

〈その２〉

第２-18図のプログラムを走らせてはみたが，レジスタに正しくデータがセットされたかをどうやって確認するのか？

以上の２点，気になりませんでしたか？　それぞれ重要な問題ですね。そこで当面の我々の課題として，この２点の問題を解決していきましょう。

```
*SD000
D000 3E-
*SD000
D000 3E-    11-AA 32-06 00-BB
D004 E0-0E 76-CC 0C-16 2C-DD
D008 20-1E FB-EE 7E-26 B9-F1
D00C 20-2E 12-F2 0C-DD 2C-21
D010 20-34 FB-12 0C-FD 20-21
D014 F0-78 25-56 ED-76 57-
*DD000,D017
D000 3E AA 06 BB 0E CC 16 DD
D008 1E EE 26 F1 2E F2 DD 21
D010 34 12 FD 21 78 56 76 57
*GD000
```

《写真８》再びカーソルが消えた

23. モニタの中身を覗く

〈その１〉の問題を解決する方法はいろいろありますが，最も合理的なのは，プログラムを終了後マシン語のコマンド・レベルに戻り，〝＊〟を表示させるのが良いでしょう。

BASICモードにあるとき

MON↗

を実行すると，N－BASICはプログラムの制御を

5C 2C番地

に移します。ここから**マシンランゲージ・モニタ**が始まります。そのしくみは次のようになっています。

① ７ＦＦＦ番地に拡張ＲＯＭが実装されており，そこに５５Ｈというデータが書かれているかチェックをします。そのときは，７ＦＦＣ番地にプログラムの制御を移します。そして７ＦＦＣ～７ＦＦＥ番地の３バイトが自由に使えますから，拡張ＲＯＭを使えば，

MON↗

でユーザーのプログラムを自由にスタートすることができます。

② そうでないときは，BASICに帰るための情報をワーク・エリアにメモし，エラーが発生したとき飛ぶ

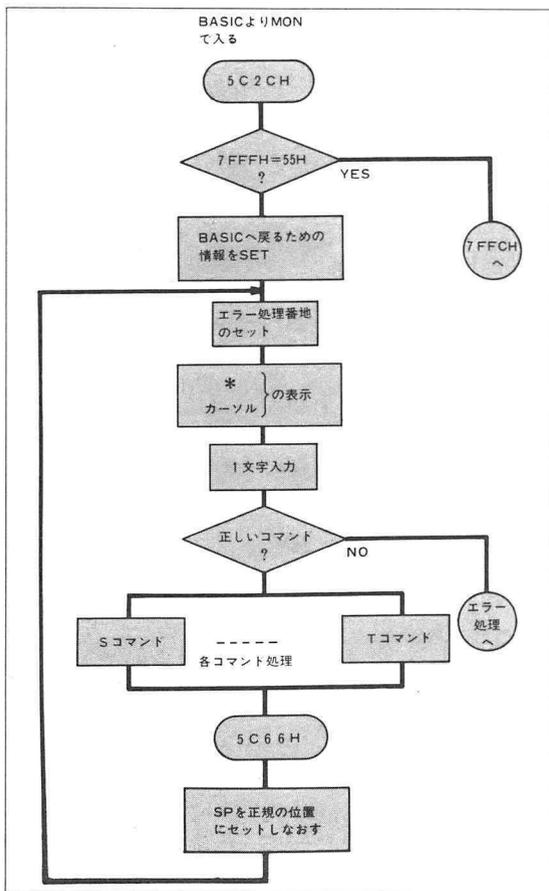
番地をセットし，〝＊〞とカーソルを表示し，マシン語のコマンド待ちとなります。

③　コマンドの1字目が入力されると，それが正しいコマンドであるかチェックし，各処理をするサブ・ルーチンに飛んでいきます。

④　各コマンドの処理が正しく終ると，

5C66番地

にジャンプします。

⑤　5C66番地からは，スタック・ポインタ（あとで学びます）を正規の位置にセットし直し，②の後半に戻ります(**第2-20図**)。

《第2-20図》モニタの構造

以上，大変難しいことを書いてしまいました。モニタの中身に関することですから，わからなくて結構です。ただ，興味が出てきたときに読み返していただければ結構だと思います。

ところで今述べてきたことで何が言いたかったのかと言えば，上記の⑤――すなわち

5C66番地

にジャンプすれば，スタック・ポインタというものを正規の位置にセットしなおし，〝＊〞に飛んでくれるということです。つまり，〈その1〉の解決の解答がここにあるのです。BASICで書けば，

GOTO　&H5C66

というような命令をマシン語で書いてやれば，

〝＊〞を表示し

マシン語のコマンド待ち

にすることができます。

24. JP命令

そのために，ここでBASICの

GOTOに相当する命令

を学ぶことにします。

> **JP（無条件ジャンプ）**
>
> **〈書式〉**　JP　*nn*（*nn*：番地）
>
> **〈機能〉**　PC←*nn*
>
> （PC：プログラム・カウンタ）
>
> *nn*番地にジャンプする。
>
> **〈マシン・コード〉**　C3

JP命令の使い方は，非常に簡単です。JPのあとに番地を書けば，その番地にポンと飛んでくれます。次にいくつか例をお目にかけますから，実験してみて理解してください。

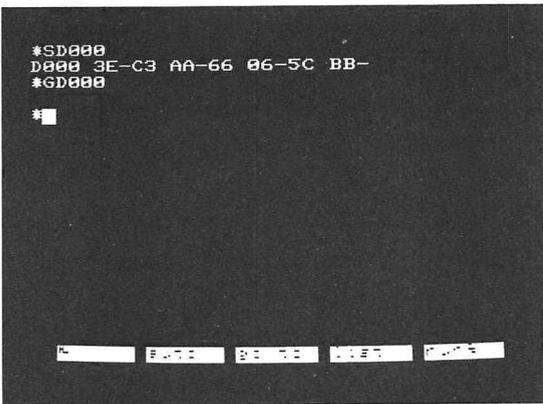
> **〈チャレンジ3〉**
>
> 走らせると，ただひたすらマシン語のコマンド・レベルに戻るプログラムを作りなさい。

このプログラムは，**第2-21図**のようになります。5C66番地にジャンプすれば良いのですね。マシン語に変換するとき，**5Cと66を入れ換える**ことに注意してください。

番　地	マシン語	アセンブリ言語
D000	C3　66　5C	JP　5C66H

《第2-21図》マシン語のコマンド・レベルに戻るプログラム

これを入力し，実行したのが**写真9**です．ちゃんと〝＊〞が表示されましたね。これで**課題〈その1〉**が

どうやら解決したようです。

《写真9》JP命令を入力，実行

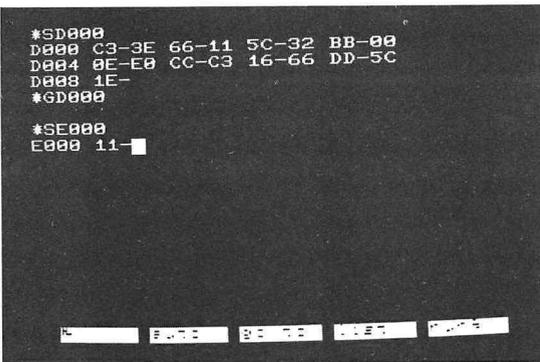
〈チャレンジ4〉
第2-12図のプログラムを，実行後マシン語コマンド・レベルに戻るようなプログラムを作りなさい。

これも簡単ですね。第2-12図のプログラムのうち，HALTを〝JP　5C66H〟に変えれば良いのです。でき上がったプログラムが，第2-22図です。またこれを実行し，ただちに

ＳＥ０００↗

でプログラムの検証をしているのが写真10です。以前のHALTを使うのに比べ，いかに楽であるかが良くお分りいただけたと思います。

番　地	マシン語	アセンブリ言語
D000	3E　11	LD　A, 11H
02	32　00　E0	LD　(0E000H), A
05	C3　66　5C	JP　5C66H

《第2-22図》プログラムの改良

《写真10》プログラムの検証

25. N-BASICのホット・スタート

せっかくＪＰ命令を覚えたのですから，もう少しこの命令を使った例をお目にかけましょう。

〈チャレンジ5〉
次にN-BASICインタプリタの中の，各種ルーチンのスタート番地を掲げてあります。それぞれにジャンプするプログラムを作りなさい。プログラムは，Ｄ０００番地から割り当てること。ただし，１本にまとめて作っても結構です。

① N-BASICコールド・スタート
　＝００００番地
② N-BASICホット・スタート
　＝０００８番地
③ N-BASICホット・スタート
　(画面をクリアせず)
　＝００８１番地

このチャレンジに登場した各番地は，覚えておくと便利ですから，ここで良く理解しておいてください。その前に，この問題文の中の言葉に馴染みのない人がいるかもしれませんので，いくつか説明しておきましょう。

インタプリタ(interpreter)
高級言語（これ自体は，ＣＰＵは理解できない）で書かれたプログラムを機械語に翻訳するプログラム。翻訳プログラムは，**アセンブラ，インタプリタ，コンパイラ**の３種類がある。そのうちインタプリタは，プログラム実行段階で逐次翻訳を進めていく方式をとる。それに対しコンパイラでは実行前に全てを機械語に翻訳してしまう。アセンブラは，コンパイラの一種である。

インタプリタ，コンパイラについての話しは，もう飽き飽きしていると思いますので，サラリと触れておきます。**マシン語**に対し，BASICのように人間の言葉に近いプログラム言語を**高級言語**といっています。しかし高級言語そのままでは，ＣＰＵは理解できません

から，プログラム実行に際しマシン語に変換する必要があります。この役目を司るのが**翻訳プログラム**です。

ところで翻訳するのに，事前にザッと全てを翻訳してしまう方式のものを**コンパイラ**，それに対し少しずつ翻訳しては実行するものを**インタプリタ**と言っています。集合の記号で書けば，

BASIC←｛X：X＝インタプリタ｝

または，インタプリタ＝｛BASIC，…｝

となります。しかし，BASICはインタプリタの代表的なものであり，その際たるものですから

インタプリタ＝BASIC

と考えても良い位です。

> **ルーチン(routine)**
> プログラムを構成するまとまった命令群。ルーチンは，メインルーチン（mainroutine）とサブルーチン（subroutine）の二つがある。

プログラムの中心となり，サブルーチン（マシン語のプログラムでは割込処理ルーチンもある）をコールする方をメインルーチン，また逆にメインルーチンから呼び出され，実行し，再びメインルーチンの中断箇所の次に返してやるのをサブルーチンと呼ぶのは，良く御存知ですね。

> **コールド・スタート(cold start)**
> プログラムを最初から実行すること。すべての初期化が行われる。
> **ホット・スタート(hot start)**
> プログラムを実行させるとき，最初の初期化の一部，また全部を省略し，プログラムの途中から実行させること。

この二つの言葉は，あまり馴染みがないかもしれません。N－BASICを例に説明してみます。

N－BASICインタプリタは，００００番地から始まっています。そこで００００番地からプログラムを走らせれば，

N－BASICをコールド・スタートさせた

ということになります。これは，**電源ONの状態**，もしくは**リセット・ボタン**を押したときに相当します。なぜならＺ－80ＣＰＵは，電源ＯＮスタート時，もしくは$\overline{\text{RESET}}$信号(リセット・ボタンを押すと発生します）を受け付けると，００００番地から実行を開始するように設計されているからです。N－BASICがコールド・スタートすると，ユーザーのプログラムや変数をクリアしたり，ＴＶ画面を40字モードにセットしたり，ファンクション・キーを設定したり等各種初期化が行われるのは，良く御存知のことと思います。

次に**ホット・スタート**について。

よくプログラムが停止状態に入ったとき，リセット・キーをそのまま押したのではそれまで作ったBASICのプログラムが消えてしまいますから，

STOPキーを押しながら

リセット・キーを押す

ということをやると思います。これが，

N－BASICのホット・スタート

です。STOPキーを押しながらリセット・キーを押すと，N－BASICの０００８番地からプログラムが再開されます。すると，ＴＶ画面やユーザー・プログラム，ファンクション・キーの設定等の初期化の一部が省略され，あなたのプログラムも破壊されることなくN－BASICが始まるわけです。

以上，コールド・スタートとホット・スタートの違い，お分りいただけたでしょうか？

26. マシン語からBASICへ

〈チャレンジ５〉の題意はおおむね理解いただけたと思いますので，その解答に移りましょう。

番　地	マシン語	アセンブリ言語
D０００	C3　00　00	JP　　0000H
０３	C3　08　00	JP　　0008H
０６	C3　81　00	JP　　0081H

《第2-23図》N-BASICのホットスタートとコールドスタート

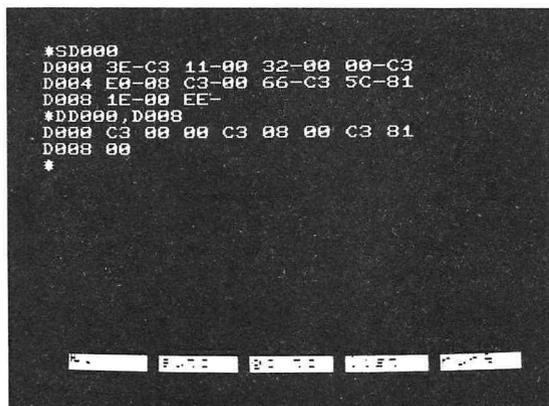
第２-23図をみてください。これは①～③を１本のプログラムにまとめたものです。**80系特有の注意**は覚えていますね。なお，プログラムの最後にＨＡＬＴ命令や，ＪＰ　５Ｃ６６Ｈは不要です。なにしろこのプログラムは１度走らせると**BASICインタプリに飛んでしまいます**から，２度とここに戻ってくることはありま

せん。

さて，このプログラムの走らせ方ですが，なにせ3本のプログラムを1本にまとめたものですから，

3通りの走らせ方

があります。順に実験してみましょう。

① **Sコマンド**でプログラムを入力し，**Dコマンド**で確認する。これは，OKですね(**写真11**)。

《写真11》Sコマンドで入力，Dコマンドで確認

② まず1つ目。

GD000↗

これで0000番地にジャンプしますから，電源ONと同じ状態が起こります。すなわち，N－BASICがコールド・スタートしたわけです。

10 PRINT "HOT START TEST OK!"

㊟ BASICのプログラムであれば、なんでも良い

《第2-24図》ホット・スタート用BASICプログラム

③ 続いてホット・スタートの実験をします。そのため，**第2-24図のBASIC**プログラムを入力してください。いちおう正しく動くか，RUNしてみてください。準備ができましたら，MON↗で"*"に移ります。

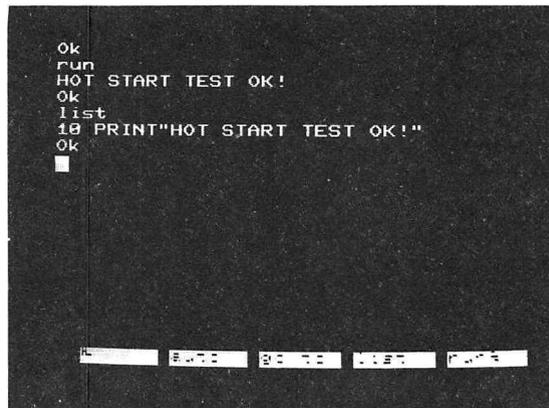
④ そして，二つ目のプログラムを走らせます。

GD00 3↗

画面がクリアされ，"OK"のメッセージが表示されますから，BASICのコマンド・レベルになったことは間違いありません。②と違って"**N－BASIC**のスタート・メッセージ"は出ません。

⑤ そこで**RUN**↗してみてください。

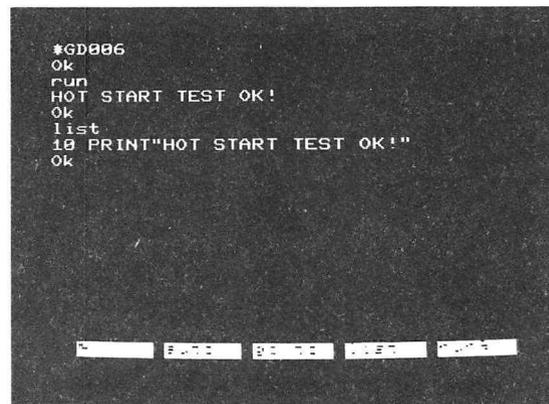
HOT START TEST OK!

のメッセージが出るはずです。**LIST**をとってみてください(**写真12**)。見事，**第2-24図**のプログラムが残っています。ホット・スタートに成功したのです。

《写真12》LISTをとってみる

⑥ 3番目は，同じくホット・スタートですが，今度は**画面をクリアしたくないとき**に使います。MON↗で"*"にし，

GD006↗

で3番目のプログラムをスタートさせます。

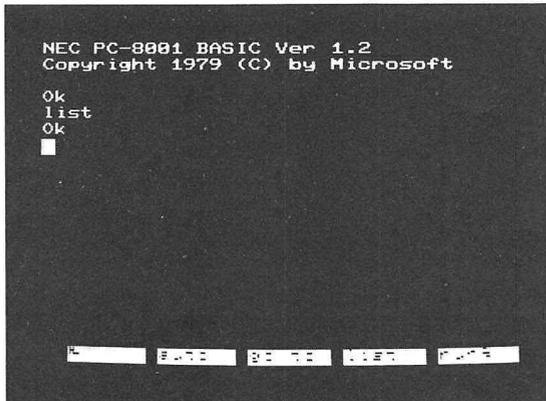
⑦ 今度は，画面がクリアされず，すぐに"OK"のメッセージが出ると思います。RUNさせると，⑤と同じことが起きます。LISTを取ると，やはり**第2-24図**のプログラムは残っていました。すなわち画面をクリアせずに**N－BASIC**をホット・スタートさせたわけです(**写真13**)。

《写真13》N-BASICホット・スタート

⑧ 最後にもう1度**MON**↗で"*"にし，

GD000↗

で1番目のプログラムを走らせてください。N－

BASICに戻ったら，LISTをとってください（**写真14**)。やはり**コールド・スタート**ですから**第2-24図**のプログラムは消滅していました。

以上で〈チャレンジ5〉の説明を終ります。

《写真14》LISTをとる

27. 課題2の解決のために

だいぶ話が横道にそれてしまいました。ここらで話しをもとに戻しましょう。

我々は，**第2-18図**のプログラムを走らせていました。ところが，気になる問題が二つ発生していました。そしてその一つ目が課題を解決するために，**ＪＰ命令**を学び，その応用例をいくつか見てきました。

そこで，今度は**二つ目の課題**を解決してみたいと思います。それは，

「**第2-18図**のプログラムを走らせた結果，レジスタの値が正しくセットされたか確認をしたい」

というものでした。この課題を解決するには，**第2-18図**のプログラムのあとに，

レジスタの値を確認するための
プログラムを追加すれば良い

のです。それは，最良の方法ではありませんが我々の知っている**知識の範囲で可能**ですから，とりあえずその方法で解決してみましょう。

ヒントは，**第2-12図**のプログラムの中にあります。このプログラムでは，

Ａレジスタ──→１１Ｈ

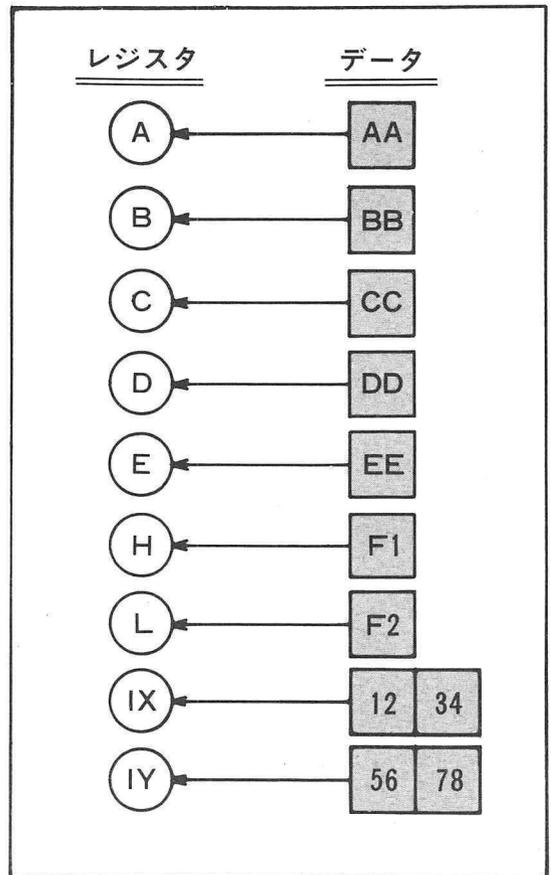
のようにセットしました。そして，Ａレジスタにデータが正しくセットされたかを確認するため，ただちに

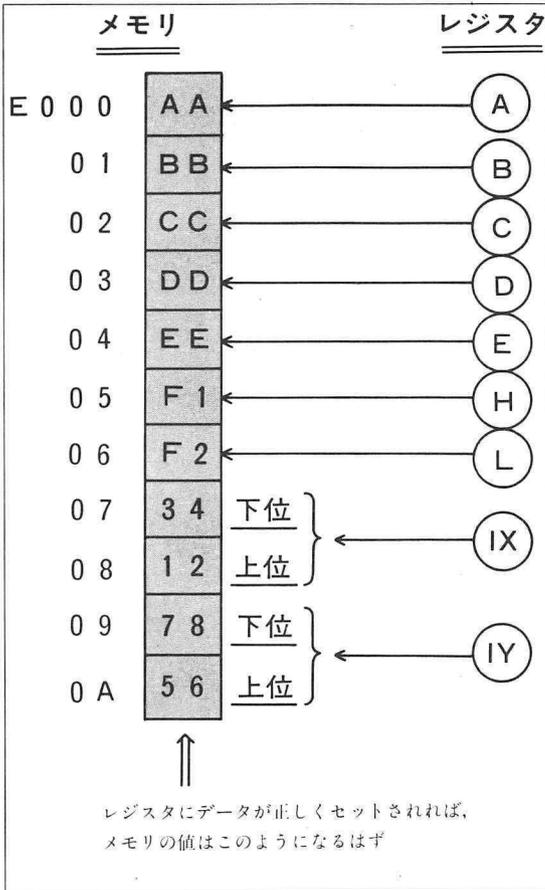
Ｅ０００番地←──Ａレジスタの値

のようにしています。そして，プログラムを走らせたあと，Ｅ０００番地を見ることでＡレジスタの値を確認しています。なぜこのような面倒な方法をとるかといえば，我々のモニタ・コマンドでは，

レジスタの中身は見られないが，
メモリの中身なら見ることが可能

だからです（Ｓコマンドなり，Ｄコマンドなりで見られますね。しかし，レジスタの中身を見るためのコマンドはありませんね？)。

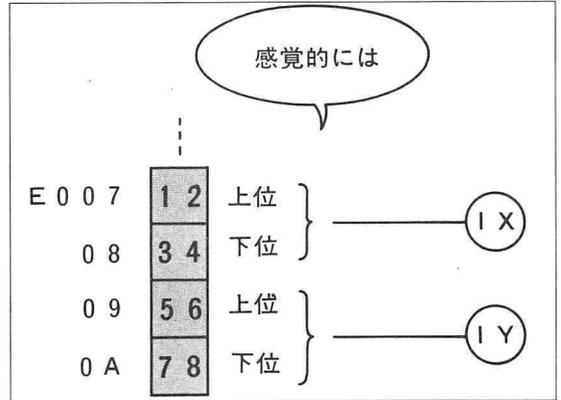
そこでこの例にならって，我々は次の方針をとることにします。

《第2-25図》第2-18図でレジスタにセットされる値

まず**第2-25図**をみてください。プログラムが正しく動けば，メモリの値はこのようになるはずです。そして，これを確認するため，これらのレジスタの値をメモリ上に移すことにします。その番地の割り付けは，**第2-26図**のように

《第２-26図》レジスタの値をメモリにおとす

Ｅ０００〜Ｅ０００Ａ番地

に移すことにします。なお，この図で**１点だけ注意**しておくことがあります。それは，ＩＸ，ＩＹの各レジスタについてです。この二つは，他のレジスタと異なり，**16ビット（＝２バイト）のデータを記憶**できるということです。したがって，この２バイトのデータをメモリに移すには，**メモリも２バイト必要**です。そのとき，この２バイトの

上位と下位のどちらを先に
メモリに移すか

が重要です。感覚的には**第２-27図**のように

《第２-27図》データの上位と下位を逆に格納する

上位のデータから順に格納——×

すればよいようにみえます。しかし，実際は

下位のデータから順に格納——○

する方が合理的です。それは学習するにしたがって明らかになるように，

80系のCPU で２バイトのデータを扱う
命令では，上位と下位が逆になっている

からです。このことは，２バイトのデータをマシン語にハンド・アセンブルするときにも経験しましたね。**第２-26図**でも，ちゃんと上位・下位が逆になっています。

以上で我々の方針は決まりました。あとは，この図を見ながらプログラムを書くだけです。

28. レジスタを見る機能を追加する

〈チャレンジ６〉

第２-18図のプログラムに，レジスタの値を**第２-26図**のように移すプログラムを追加しなさい。

さあ，これくらいのプログラムになるとだいぶ長くなります。すこし説明が必要になるでしょう。初めての方には，少し難しいかもしれません。ゆっくり味わってください。

先に完成したプログラムを見ていただきましょう。

```
D000 3EAA      LD   A,0AAH    ┐ここまでは第2-18図
D002 06BB      LD   B,0BBH    │のプログラムと同じ
D004 0ECC      LD   C,0CCH    │
D006 16DD      LD   D,0DDH    │
D008 1EEE      LD   E,0EEH    ┘
D00A 26F1      LD   H,0F1H
D00C 2EF2      LD   L,0F2H
D00E DD213412  LD   IX,1234H
D012 FD217856  LD   IY,5678H
D016 3200E0    LD   (0E000H),A  ┐
D019 78        LD   A,B         │
D01A 3201E0    LD   (0E001H),A  │
D01D 79        LD   A,C         │
D01E 3202E0    LD   (0E002H),A  │
D021 7A        LD   A,D         │
D022 3203E0    LD   (0E003H),A  │レジスタの値を
D025 7B        LD   A,E         │メモリに移す
D026 3204E0    LD   (0E004H),A  │
D029 7C        LD   A,H         │
D02A 3205E0    LD   (0E005H),A  │
D02D 7D        LD   A,L         │
D02E 3206E0    LD   (0E006H),A  │
D031 DD2207E0  LD   (0E007H),IX │
D035 FD2209E0  LD   (0E009H),IY ┘
D039 C3665C    JP   5C66H      * へジャンプさせる
```

《第2-28図》レジスタ表示プログラムを追加する

第2-28図がそうです。

① Ｄ０００～Ｄ０１５番地については，**第2-18図**のプログラムとまったく同じです。ここは，問題ありませんね。

② 次に最後のＤ０３９番地をみてください。

　　　ＪＰ　５Ｃ６６Ｈ

となっています。これは覚えていますね。ＨＡＬＴの変わりにプログラムの最後に置くものです。マシン語のモニタにジャンプさせるのでしたね。

③ 残りの真中にはさまれたＤ０１６～Ｄ０３８番地が，〝**レジスタの値をメモリに移している**〟部分です。

④ そのうちのＤ０１６番地

　　　ＬＤ　（０Ｅ０００Ｈ），Ａ

については，すでに**第2-12図**でやりましたのでおわかりでしょう。Ｅ０００番地にＡレジスタの値を移しているのですね。

⑤ 残りの部分が未知の部分です。ここについては，節を改めて御説明することにします。

29. アキュームレータ

まず8ビット・レジスタの方から見ていきます。最初は，**Bレジスタ**について，おそらく大部分の方は，〝Ａレジスタと同様にやればよい〟と考え，

　　　ＬＤ　（０Ｅ００１Ｈ），Ｂ———①

としたのではないでしょうか？　たしかに鋭いところをついています。しかし，残念ながらこれは

　　　間違い！

です。

理由は簡単です。〝**Z－80活用表**〟をみてください。〝８ビット〟の上から13段目，

　　　ＬＤ　（nn），Ｘ

のところを右にずっとみていってください。Ａレジスタ以外のところには，**マシン語の記入がありません。**これは，Ａレジスタ以外には，そのような命令がないからです。したがって，いくらアセンブリ言語で①の命令を書いても，それをハンド・アセンブルすることはできません。

以上が①の使えない理由です。

今見ましたように，

　　　Ａレジスタは他のレジスタに比べ
　　　命令が強化

されています。とくにあとで出てくる計算（Z－80のマシン語では，加減算しかできませんが）では，Ａレジスタの独壇場で，他のレジスタはＡレジスタの介入なしには計算できません。

これは，Ａレジスタのもつハード上の特性からきています。なぜなら，Ａレジスタは

　　　アキュームレータ

という特別なレジスタだからです。

> **アキュームレータ(accumulator)＝累算器**
> 演算やデータの入出力の中心となるレジスタで，演算の結果はアキュームレータに入る。

80系のＣＰＵでは，アキュームレータもレジスタと呼んでいますが，68系のＣＰＵでは，ハッキリ〝**アキュームレータ**〟と〝**レジスタ**〟を区別しています。ちなみに6800ＣＰＵでは，アキュームレータが二つあります。しかし，レジスタは8080ＣＰＵの方が豊富です。

30. 8ビット・レジスタの値をメモリへ

さて，**Aレジスタは他のレジスタに比べ，特権的な力を持つ**ことはわかりましたが，それだけでは①の解決にはなりませんね。そこで発想の転換をはかりましょう。

Bレジスタがダメなら

Aレジスタに置きかえてやればよい

のです。どういうことかといえば，D019番地をみてください。

LD A，B————②

と書いてあります。これも〝LD命令〞ですから，右側の値が左側に移ります。すなわち

〈Aレジスタ〉←—〈Bレジスタの内容〉

のように移す命令です。つまり②の命令を実行するとBレジスタの内容BBHがそっくりAレジスタに移されます。続いてこのAの値を

LD (0E001)，A

でE001番地に移してやれば，①の機能を実現できることになります。もう1度まとめれば，次の3段階でメモリに移しています。

Bレジスタ
⇓
Aレジスタ
⇓
E001番地

Cレジスタについても，同様にして

LD A，C
LD (0E002)，A

でメモリに移せます。プログラムのD016～D02D番地をみてください。8ビットのレジスタについては，すべて

Aレジスタを介して，レジスタの値を
メモリに移している

のがわかります。

31. 16ビット・レジスタの値をメモリへ

最後に16ビット・レジスタIX，IYの値をメモリに落とすことを考えましょう。

IX，IYレジスタは，どちらもAレジスタのように，直接メモリに代入することができます。その様子をIXレジスタを使って説明致します。〝**Z－80活用表**〞をみてください。16ビット・レジスタの上から8段目です。

LD (*nn*)，X

という命令があります。これを使えば，

LD (0E007H)，IX

とすれば良いことがわかります。またこれをマシン語に変換すると，

```
 DD  22  07  E0
LD (nn), IX
       E007番地
```

となります。なお，このマシン語を実行すると，

E007番地←—IXレジスタの下位
E008番地←—IXレジスタの上位

のようにメモリに格納されます。ちょうど**第2-26図**と対応していることに注目してください。

IYレジスタについても，同様にしてレジスタの値をメモリに落とすことができます。

32. 成　功

さあ，これで**第2-28図**のプログラムについて全て理解できました。さっそくプログラムを走らせることに致しましょう。

入力は，**Sコマンド**です。長いプログラムですから慎重にキー・インしてください。**Dコマンド**による確認――OKですね？

GD000↗

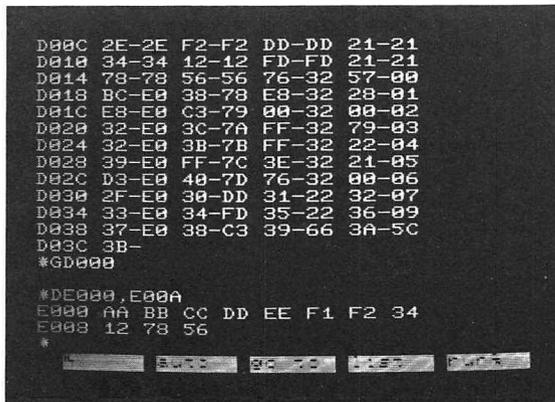
でプログラムRUNです。今度は，HALT命令を使っていませんから，すぐに〝*〞が表示され，コマンド待ちになったと思います。

次が大切です。このプログラムのミソは，レジスタの値を確認できることにあります。レジスタの値は，

E000～E00A番地

に入っています。したがって，

DE000，E00A↗

とすれば，確認ができます（**写真15**）。**第2-26図**と照らし合わせてみてください。当初の我々の設計通りになっていることがわかります。そして，これで二つの課題が解決されたことになります。

《写真15》DE000，E00Aで確認

〈第3章のおわりに〉

本章において初めて我々は，

自分で理解したプログラム

を走らせることに挑戦しました。そこから出発した我々はN－80の全レジスタを眺めました。そして主として，

レジスタにデータを代入するプログラム

に挑戦してきたわけです。するとそこには二つの課題が潜んでいました。本章の後半は，その二つの課題の解決が主体となりました。そして，我々の知識の範囲内で

レジスタの中身を知るためのプログラム

に臨んだわけです。その結晶としてでき上がったのが，**第2-28図**のプログラムです。たしかにこのプログラムによって，レジスタの値は確認できました。しかし，このプログラムにも問題がないわけではありません。あなたは，**第2-28図**のプログラムを作成，あるいは実行してみてどのようにお感じになりましたか？

①プログラムの作成が面倒だ

②プログラムが使いずらい

- レジスタの値を見るのに，いちいちDコマンドを使うのは面倒
- しかも，見ずらい。

同感です。たしかに**第2-28図**のプログラムには，ムダな部分がたくさんあります。しかし，このことについては，あなたのマシン語に対する学習の進行に従って，自然に解決するでしょう。しかし，②の解決のためにはどうしても**マシン語による**

画面表示

の仕方を知る必要があります。次ブロックは，そのあたりから出発していくことになります。

第3ブロック

マシン語プログラミング実践2

まだまだ　ま・し・ん・ご・ぷ・ろ・ぐ・ら・み・ん・ぐ

第1章 画面表示への2大要素

〈はじめに〉

あなたが本書に挑まれたきっかけは，おそらくマシン語のマスターを願ってのことと思います。そして，その前提にはいろいろな目的があると思います。それは人によって千差万別でしょうが，最終的にはその実現手段として

TV画面の制御

が必要になってくるでしょう。そして今やそのマスターのための基礎準備は完了しています。それに挑戦する時がきたのです！

本章では，その入り口として

マシン語による画面表示への

手掛りをつかむ

ことがメイン・テーマになります。本章を読んでいくと，あまり**“演習”**が用意されていないことに気づくでしょう。その代わりたくさんの**実験**が用意されています。それは，次の理由によります。

本書では，**“TV画面の制御”**にかなりの重点をおいています。それは，**マシン語による各種制御の基礎**になると考えられるからです。したがってそれを肌で感じていただくため，本章ではあまり

知識を伝授する

という方法はとらず，

自分でその方法を体得

していただこうという方針を取っています。そのため**実験**を重視し，

〈実　験〉を通して
↓
〈仮　説〉を立て，その仮説を
↓
〈検　証〉する

という**“科学的アプローチ”**の方法を試みています。

どうぞその主旨を理解いただき，実験を繰り返していただきたいと思います。本章の終る頃には，**“画面制御の問題点”**が明白になっていることでしょう。

1. 予期せぬできごと

まず，本章はサンプル・プログラムの実行から出発しましょう。**第3-1図**をみてください。

番　地	マシン語	アセンブリ言語
D000	3E E9	LD A, 0E9H
02	32 C8 F8	LD (0F8C8H), A
05	C3 66 5C	JP 5C66H

《第3-1図》サンプル・プログラム

最初にプログラムを解読してみてください。すぐにわかると思います。

1行目：AレジスタにE9Hというデータをセットする。

2行目：Aレジスタにデータが正しくセットされたかを調べるため，Aレジスタの値をメモリのF8C8番地に移す（プログラム実行後F8C8 番地をみれば，Aレジスタの値を確認できます)。

3行目：モニタにジャンプする。

このプログラム，どこかで見たことがありますね。そうです。第2ブロックでやった**第2-12図**のプログラムを，少し変えただけです。

このプログラムでは，実行価値はない――と思われたかもしれません。しかし，プログラミングの世界では，

「たぶん，こうなるだろう」

という予測だけではダメで，**必ず実行してみる**必要があります。プログラムの実行には，必ず**たくさんの外的要件**が伴い，思わぬ落とし穴に遭遇することがあります。

それでは実行します。まず，あなたのマシンを

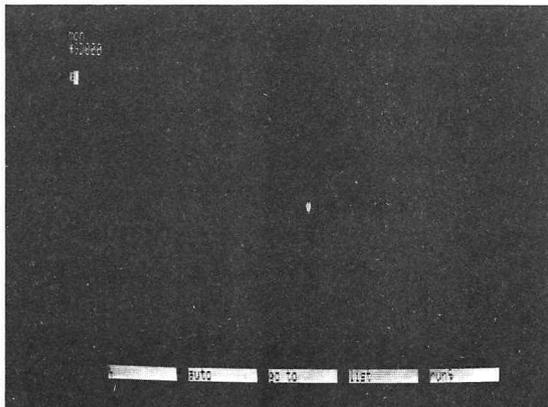
1 行80字モード

にしてください。もちろん，これはBASICで結構です。続いて，**MON**↗で“＊”，**Sコマンド**で入力，**Dコマンド**で確認――これは，いつものセオリー通りです。

GD０００↗

でRUN（**写真1**)。いかがですか？ やはり我々の予想外のことが起こりましたね。

画面の中央に♥

が出現したではありませんか！

《写真 1》♥マーク出現

2. 違いはどこだ？

第3-1図のプログラムと**第2-12図**を比べ，どこが異なるのでしょうか？ 実行結果は，まるで違います。一方のプログラムでは何も起こらず，一方のプログラムでは“♥”が表示されました。

プログラムを見比べて明らかに異なるのは，３行目の“HALT”と“JP　5C66H”。しかし，これはプログラムの停止の仕方の違いであって，プログラムの本質とは別のものです。となると，残りは１，２行目だけ。ところがここを見比べても，**アドレスとデータの値**が異なるだけです。

もう既に気がつかれた方がいるかもしれません。この

アドレスとデータの値

が異なることが大きなポイントです。とくにアドレスの値

F8C8番地

に注目してください。

我々は，前ブロックにおいてハンド・アセンブルを学びました。その際，マシン語のプログラムをメモリに割り当てるには，ある**特定の制限**があったことを思い出してください。すなわちメモリ上には，**“マシン語のプログラム可能領域”**というのがあり，それは

8020番地 }
C020番地 } **～E9FF番地**――――①

でした。**第3-1図**におけるF8C8番地は明らかに①の領域をはみ出しています。

F8C8番地は，システム(N-BASICインタプリタ)が使う場所です。**第3-1図**のプログラムではむりやりその場所にAレジスタの値を入れてしまったため，おかしなことが起こったものです。実はこの辺に**画面表示のためのヒント**が隠されているのです。一体，F8C8番地とは，何なのでしょうか？ そこらあたりにメスを入れていくことに致しましょう。

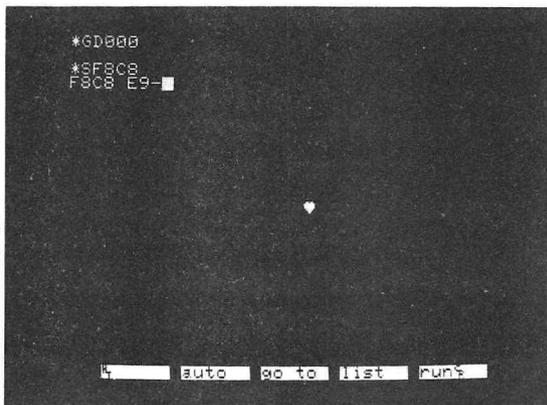
3. トランプの実験

さて，予期せぬできごとは発生しましたが，いちおう**第3-1図**のプログラムは走らせたわけです。せっかくですから，きちんと最後まで確認をしてみましょう。

SF8C8↗

とキー・インしてください(**写真2**)。ちゃんと

E9――Aレジスタにセットした値

がストアされているのがわかります。プログラムは，うまく動いたことになります。

《写真 2》ＳＦ818とキー・イン

さあ，そこで**第3-1図**のプログラムについていろいろ実験してみることにします。次の実験をためしてみてください。

〈実　験〉

① BASICモードで

CONSOLE, 1↗

を実行する。ただし，画面モードは，1行80字モードにしておいてください。

② MON↗で“＊”へ

③ SF8C8↗

E8（これ以外のキーは，押さない）

④ これ以降は指定の英数字やキーのみを押してください。

DEL（←デリート・キーのこと）

EA

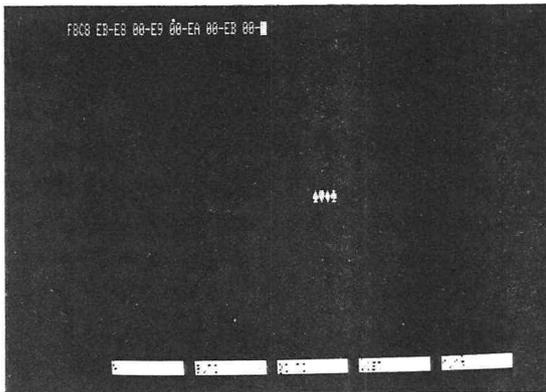
⑤ DEL

EB

⑥ DEL

E8

⑦ E9
EA
EB　}次々にキー・インする

実験は，以上です。いかがでしたか？　実験終了時の画面の様子を**写真3**に示します。何と，**トランプのマークが勢揃い**しました。

《写真3》トランプのマークが勢揃い

4. 実験から仮説をたてる

それでは，順に解説していきます。

①は，画面のスクロール範囲を上1行に限定しています。これはBASICのコマンドですから，特に問題はないですね。

③を実行すると，今まで表示されていた

に変わります。続いて次のように変化していきます。

④：♠⟶♦

⑤：♦⟶♣

⑥：♣⟶♠

⑦については，1バイトキー・インするたびに，

の順にトランプのマークが現われます。

以上の点を通して，次の2点に気がつかれたのではないでしょうか？

F8C8番地の附近＝画面表示の位置に関係ありそうだ！

書き込んだ値＝画面表示の内容に関係ありそうだ！

あなたの想像は，

ズバリ，的中！

しています。実験結果から**仮説**も立ち，どうやらここらで**画面表示のまとめにはいる**時期がきたようです。

5. TV画面用ワーク・エリア

我々のマシンのシステム・ワーク・エリアのうち

F300～FEB7番地―――①

（全3000バイト）

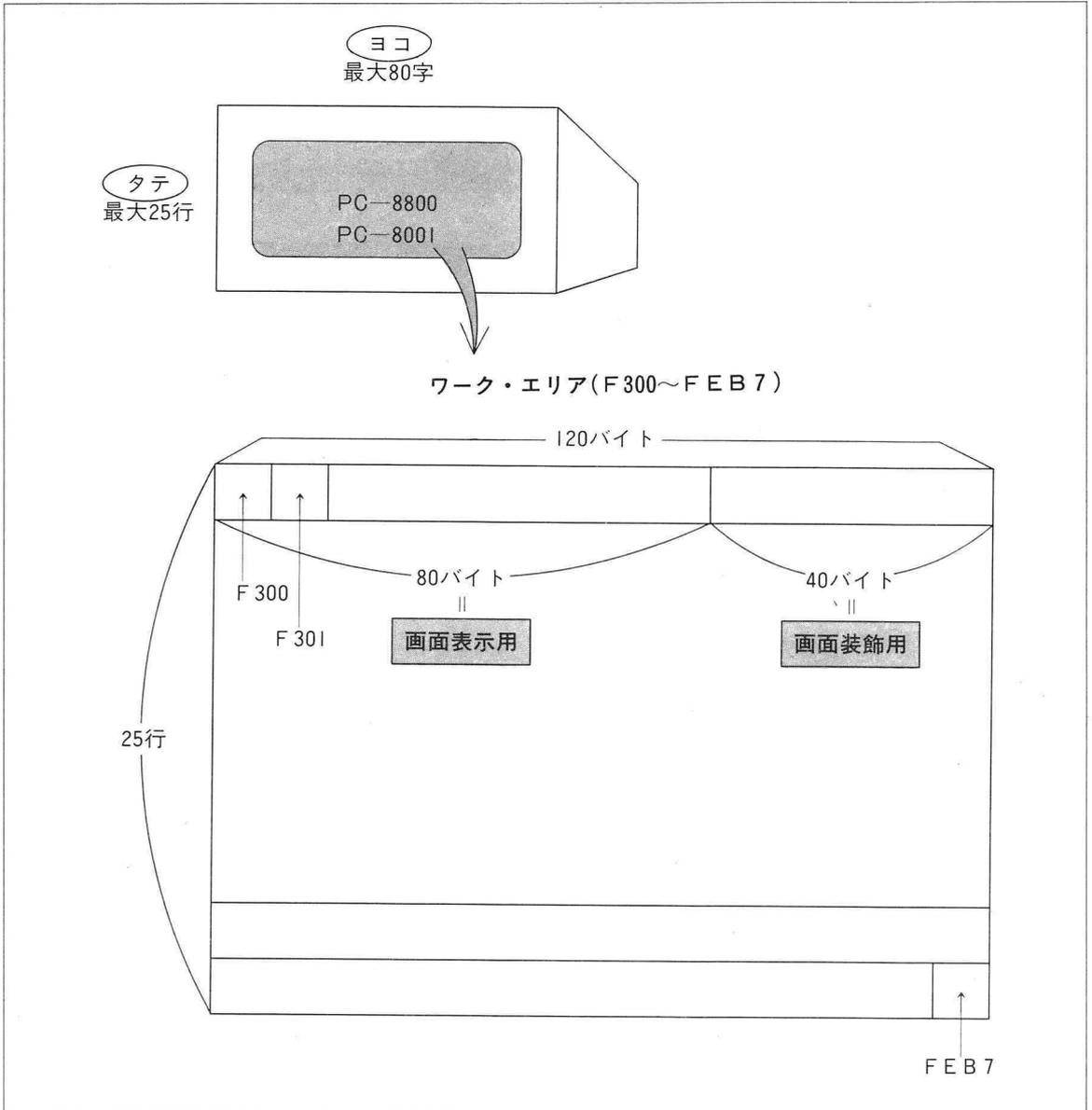
は，**画面表示**に使われています。我々のマシンは，TV**画面が25行表示**できます。そのため①も25の領域に分割されていて，**各行120バイト**ずつ割りあてられています。すなわち最初の120バイトが1行目の表示に使われ，次の120バイトが2行目の表示――という具合です。したがって

120×25＝3000（バイト）

が割り当てられているわけです。

次に**各行の構成**をみてみると，最初の80バイトがTV画面の各キャラクターに対応しています。我々のマ

シンは1行に80字まで表示できますね。したがってその80字に80バイトが対応しているわけです。すると120バイトのうち40バイトがあまりますね。この40バイトが，**画面の装飾**（たとえば色をつけたり，ブリンクさせたり）に使われています。

以上の様子を図示したのが，**第3-2図**です。この図を良くみて，もう1度これらの説明を読み直してください。まだ具体的には，どういうことかわからないと思います。しかし，自分なりのイメージを持ちながら次に進んでいただきたいと思います。

《第3-2図》画面表示用システム・ワーク・エリア

以上のように，画面の装飾関係を無視すれば，メモリ上の①の部分は，画面の表示と1対1対応しています。たとえば次のようになっています。

LOCATE　0，0⟶F300番地
LOCATE　1，0⟶F301番地
LOCATE　2，0⟶F302番地
⋮

(注) これは，白黒モードで1行80字のときです。

そこで，このことを実験で確めてみましょう。

① BASICのコマンド・レベルで

CONSOLE　24, 1↗

WIDTH　80↗

を実施する。これで画面が1行80字モード，またスクロール範囲がTV画面の下1行に限定されます。

② MON↗　で“＊”へ。

③ SF300↗

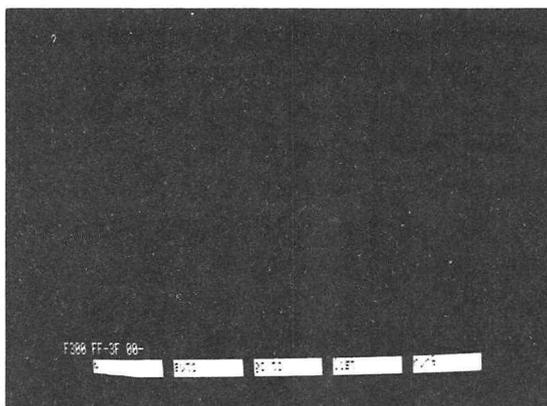
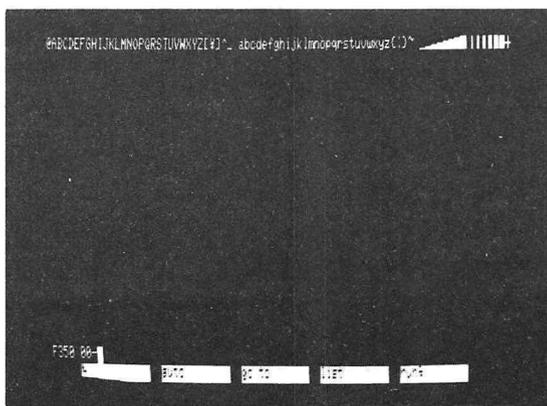
これでデータを書き込むアドレスがF300番地にセットされます。すなわち画面の左上に表示する準備ができたわけです。ここに何でもいいからデタラメなデータを書き込んでください（16進数をキーインしてくださいという意味です）。たぶん何かが画面の左上に表示されるはずです（写真4）。なお不幸にして何も表示されなかった人も，別に気にする必要はありません。次の実験に進んでください。

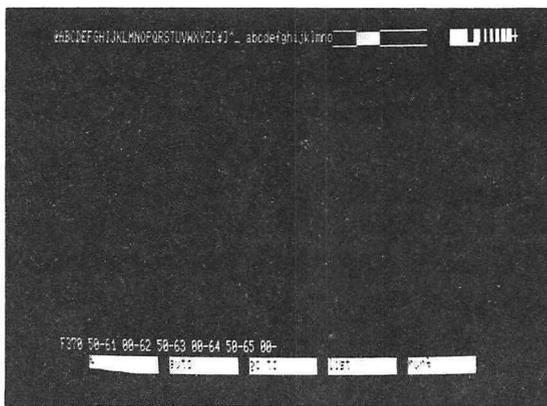
④ 今は，F301番地の書き込みになっています。ここにも何か書き込んでください。今度は③の右隣に何か表示されるはずです。

⑤ 以上の操作を繰り返し，とにもかくにも，1行目の最後まで何かを表示させてみてください。F34F番地がちょうど80字目にあたり，次はF350番地になります（写真5）。

⑥ ここから40バイトは，**画面の装飾**に使われるのでしたね。ここにもかまわず適当な16進数を入力していってください。画面がブリンクしたり，リバースしたり，いろいろなことが起こるでしょう（写真6）。

⑦ それでもガチャガチャ，キーインしていくとやがて40バイトを使いきり，画面の2行目に到達致します。アドレスはF378番地をさします。

⑧ そろそろTV画面用のシステム・ワーク・エリアの意味がわかりかけてきたと思います。したがって，あとはどんな実験をするかあなたにおまかせします。

《写真4》画面左上に？マーク表示

《写真5》1行目の最後まで表示させる

《写真6》画面がブリング，リバースしたりする

6. “どこに？”表示する

前節の実験により，TV画面に表示するには，

表示したい位置のワーク・エリアの番地――①

を求め，

そこに表示したいデータを書き込む――②

ことにより実現できそうだということがおわかりになったと思います。

①については，付録の“**レイアウト・シート**”をみてください。“**80×25モード**”と“**40×25モード**”の

2種類がありますが，とりあえず標準の**“80×25モード”でマスター**してください。**“40×25モード”はそれを1バイトおきに使うだけ**ですから。

そういうわけで“80×25モード”の“レイアウト・シート”をみてください。あなたが，TV画面の任意の位置に**何か**（この**“何か”**については，いまは伏せておきます。つまり前記の②の部分ですね。これについては後述するとして，いまは①の**“どこに？”**をマスターしてください）を表示したいとき，次の手順をふめば，それを実行することができます。

〈TV画面表示への手順〉

① 例として，ヨコ10番目，タテ3番目の位置に**何か**を表示してみましょう。つまり，

LOCATE　9，2

の位置です。

② 付録の“レイアウト・シート”により，ワーク・エリアの番地を求めます。この場合

F3F9番地

です。

③ SコマンドでF3F9番地に表示したいデータ（この場合，デタラメな16進数）を書き込む。

ここまで**“どこに？”**については，OKですね？

次は，“何を？”です。

7.“何を？”表示する

付録の**“PC-8001キャラクター・コード表”**をみてください。この表，どこかで見たことがありませんか？　あるはずです。あなたのマシンの**“ユーザーズ・マニュアル”**に同じものが載っています。BASICを使うだけでしたら，この表はとくに必要ありません。しかし，これからマシン語を使おうとするあなたにとっては，この表が**必需品**になるでしょう。なにしろ前節まで伏せておいた表示のための**“何を？”**はこの表の中にあるのですから。

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		D_E		0	@	P		p	▁	┴		ー	タ	ミ	═	╳
1	S_H	D_1	⁝	1	A	Q	a	q	▂	┬	。	ア	チ	ム	╞	円
2	S_X	D_2	‖	2	B	R	b	r	▃	┤	「	イ	ツ	メ	╪	年
3	E_X	D_3	#	3	C	S	c	s	▄	├	」	ウ	テ	モ	╡	月
4	E_T	D_4	$	4	D	T	d	t	▅	─	、	エ	ト	ヤ	◢	日
5	E_Q	N_K	%	5	E	U	e	u	▆	─	・	オ	ナ	ユ	◣	時
6	A_K	S_N	&	6	F	V	f	v	▇	│	ヲ	カ	ニ	ヨ	◥	分
7	B_L	E_B	▼	7	G	W	g	w	█	│	ァ	キ	ヌ	ラ	◤	秒
8	B_S	C_N	(	8	H	X	h	x	▏	┌	ィ	ク	ネ	リ	♠	
9	H_T	E_M	)	9	I	Y	i	y	▎	┐	ゥ	ケ	ノ	ル	♥	
A	L_F	S_B	*	:	J	Z	j	z	▍	└	ェ	コ	ハ	レ	♦	
B	H_M	E_C	+	;	K	〔	k	{	▌	┘	ォ	サ	ヒ	ロ	♣	
C	C_L	→	,	<	L	¥	l	:	▋	╭	ャ	シ	フ	ワ	●	
D	C_R	←	−	=	M	〕	m	}	▊	╮	ュ	ス	ヘ	ン	○	
E	S_O	↑	.	>	N	^	n	~	▉	╰	ョ	セ	ホ	゛	╱	
F	S_I	↓	/	?	O	_	o		┼	╯	ッ	ソ	マ	゜	╲	

《第3-3図》PC-8001キャラクタ・コード表

そこで“**キャラクター・コード表**”のひき方を説明しておきましょう。

<“キャラクタ・コード表”のひき方>

① 例として，これから英文の大文字“**X**”を表示したいとします。

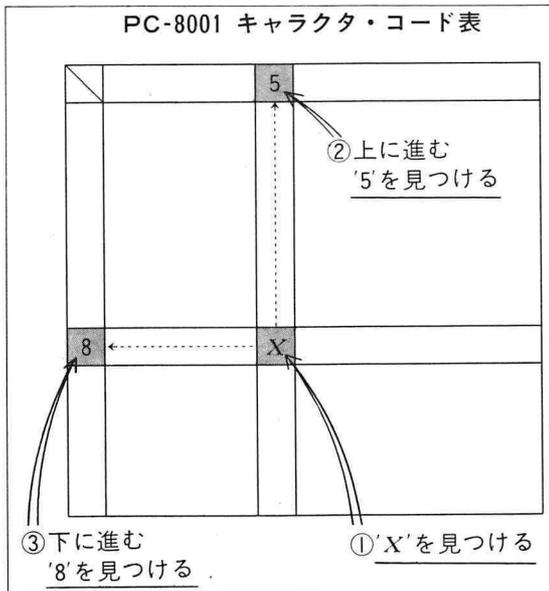
② Xが“キャラクタ・コード表”のどこにあるか捜します。真中やや左寄りのところにありますね。

③ 見つけたら，その列の１番上を見ます。５と書いてありますね。

④ 次に行の左側を見ます。８と書いてありますね。

⑤ ③，④を組み合わせた

58H

が求める“X”のキャラクタ・コードです(**第3-4図**)。

《第３-４図》キャラクタ・コードを見つける

“キャラクタ・コード表”のひき方については，お分かりいただけましたね。もう二～三の例をあげておきましょう。

<表示したいもの>	**<キャラクタ・コード>**
¥	→5CH
■	→87H
ツ	→C2H
♥	→E9H

8. 二要素のドッキング

以上で，“**何を**”と“**どこに**”が概ね明らかになりました。あとは，この二つをドッキングさせれば，我々は，

マシン語でTV画面を自由に
あやつれる

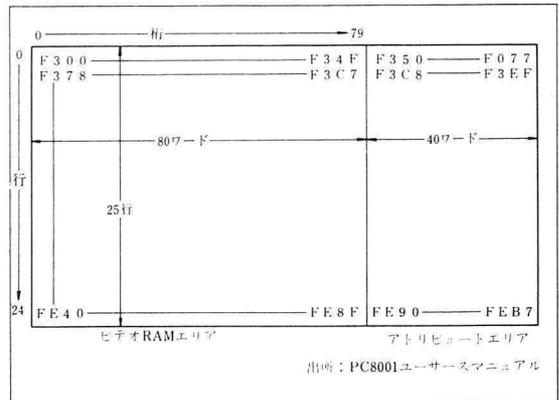
ようになります。

第3-5図をみてください。我々は，画面表示用のワーク・エリアの意味を知っています。そのワーク・エリアは，大きく二つの用途に使われていました。一つは，**画面の表示**に関する部分で，普通

ビデオRAM

と呼んでいます。もう一つは，**画面の装飾**のために使われる部分です。こちらは，

アトリビュート・エリア

と呼んでいます。

《第３-５図》ビデオRAMの構成

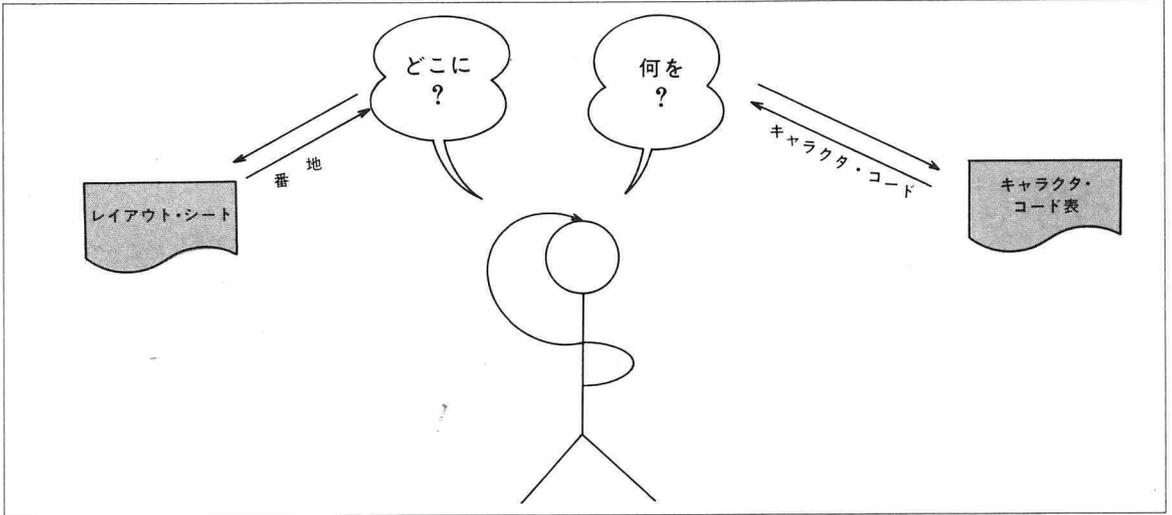
第3-5図は，“ユーザーズ・マニュアル”のビデオRAMのところです。既にあなたは，この図の意味を理解できますね。

<チャレンジ1>

LOCATE　10，10

（１行80字モード）

の位置に●を表示しなさい。

画面表示についてのこれまでの知識をまとめると，**第3-6図**のようになります。

《第 3 - 6 図》画面表示のための 2 つの要素

そこでこの〈チャレンジ〉を考えてみましょう。画面表示するには，**二つの要素**がありましたね？

① **どこに？**

"レイアウト・シート" を使って**ビデオRAM** のアドレスを求めます。

LOCATE 10, 10──→F7BA番地

② **何を？**

"キャラクタ・コード表"を使って，キャラクタコードを求めます。

●──→ECH

これで表示の準備ができました。

SF7BA↗

EC↗

こうすることにより，BASICの

LOCATE10, 10

PRINT "●"

と同じ現象が起こるはずです。

以上，我々は画面表示のための 2 要素

どこに？

何を？

をドッキングさせることで，好きなキャラクタを，好きな場所に表示できるようになったわけです。

しかし，──。

〈第1章のおわりに〉

本章により第 3 ブロックがスタートしました。本ブロックのターゲットは，

画面表示への糸口を探る！

ことにありました。画面表示を自由に行うためには，

何を？

どこに？

を自由に制御する必要があります。本章は，主としてこの二要素の解決にあてられてきたわけです。そして，それはひとまず解決したかのように見えました──。

たしかに我々は，その二要素の制御法を明らかにしました。しかし，鋭いあなたは次の点に気がつかれたでしょう。すなわち，

我々は二要素を制御したのち

画面表示をする手段に

モニタ・コマンドを使用していた！

Sというモニタ・コマンドを使ってそれを実現していたのです。これでは，**プログラムによって画面表示をしたことにはなりません。**我々がやりたいのは，それを

マシン語のプログラムの中で

実現したかったのです。これで次章の目標は決まりました。イザ。

第2章
画面表示への実現

〈はじめに〉

前章における種々の実験により，我々は

画面制御への糸口

をつかみました。いよいよ本章で

画面制御の実現

に取り組むことになります。その基本として最初は，**“1キャラクタを画面の自由な位置に表示する”**ことを試みます。続いてそれを演繹していき，**“5キャラクタ”**，**“80キャラクタ”**と拡張していきます。その過程において，**“ループ”**，**“相対番地”**等の**新しい概念**が次々に登場してきます。最初はやはりこれらの概念は馴染みにくいかもしれません。したがって，これらの概念の説明にはかなりのページ数をさく予定です。

いずれにしても本章が，

マシン語修得への一つ関所

になることは間違いないと思います。無事に本章を通過されるよう期待しています。

9. 1キャラクタの表示

前章までの知識を利用し，マシン語による画面表示を，

プログラムの中で実現

してみましょう。その基本となるのは，次の演習です。

〈チャレンジ2〉

```
LOCATE 40, 10
PRINT "♥"
END
```

のプログラムを，マシン語で実現しなさい。

この問題は，いわゆる

1キャラクタを自由な位置に表示

しようとするもので，**画面制御すべての基礎**になります。ここでは，レジスタのいろいろな使い方を覚えるため，いろいろな方法でアプローチしてみます。

しかし，いずれの方法で挑むにしても，まず必要なデータを用意しておく必要があります。

① **アドレス（番地）**

LOCATE 40, 10

の座標を**“レイアウト・シート（80×25）”**によりアドレスを求めます。

F7D8番地

が概当します。

② **キャラクタ・コード**

“キャラクタ・コード表”により，“♥”の キャラクタ・コードを求めます。

E9H

が求めるコードです。

以上でプログラミングの準備はできました。最初は，今までの知識のみでこなしてみましょう。例のAレジスタを使う方法です。**第3-7図**が，そのプログラムです。

番　地	マシン語	アセンブル言語
D000	3E E9	LD A, 0E9H
02	32 D8 F7	LD (0F7D8H), A
05	C3 66 5C	JP 5C66H

《第3-7図》既知の範囲で作る

① まず1行目，Aレジスタに“♥”のキャラクタ・コードをセットします。

② それを2行目でF7D8番地に移しています。これは，前ブロックにおける

SF7D8↗

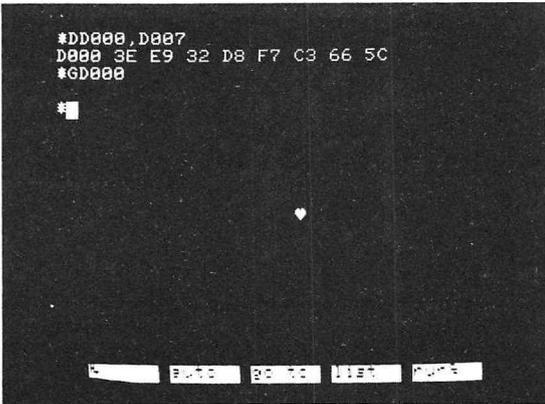
E9

に相当します。

③ 3行目は，“＊”にジャンプさせてプログラムを終了させています。

以上は，簡単に理解できたと思います。それでは，さっそくこのプログラムを走らせてみることにしましょう。やり方はわかりますね？

① **Sコマンド**でプログラムを入力する。
② **Dコマンド**で確認する。
③ **Gコマンド**でプログラムを走らせる。

以上の様子を示したのが，**写真7**です。

《写真7》 Gコマンドでプログラムを走らせる

10. インデックス・レジスタを用いて

次には，〈チャレンジ2〉を**インデックス・レジスタ**を使って実現する方法を紹介します。Z-80には，二つのインデックス・レジスタがありました。

IXレジスタ
IYレジスタ

の二つですね。前者を用いたプログラム例が，**第3-8図**，また後者を用いたのが**第3-9図**です。どちらも方法は同じですから，ここではIYレジスタを使った方を説明することにします。

番　地	マシン語	アセンブル言語
D000	DD 21 D8 F7	LD IX, 0F7D8H
04	DD 36 00 E9	LD (IX+00H),0E9H
08	C3 66 5C	JP 5C66H

《第3-8図》 IXレジスタを使って

番　地	マシン語	アセンブル言語
D000	FD 21 D8 F7	LD IY, 0F7D8H
04	FD 36 00 E9	LD (IY+00H),0E9H
08	C3 66 5C	JP 5C66H

《第3-9図》 IYレジスタを使って

マシン語のプログラムを組んでいると，**あるメモリを中心にその前後とデータのやり取りをする**ということがよく出てきます。そんなとき，このインデックスレジスタを使うと便利ですから，ここでその使い方を理解しておくと良いでしょう。

そこで，IYレジスタを使って，メモリにデータを転送する方法です。一般的には

d：変位（1バイトの数）
n：データ（1バイトの数）

LD (IY+d), n

の形を用います。ここでnは，転送したいデータで，〈チャレンジ2〉の場合でしたら♥のキャラクタ・コードE9Hのことです。**第3-9図**の2行目もそうなっています。残りのIYとdが，少し理解しにくいと思います。普通，**この命令を実行する前に，IYを希望の値にセットしておきます**。たとえば，**第3-9図**のプログラムでしたら，1行目の

LD IY, 0F7D8H

で，

IY=0F7D8H

にセットされています。そして，その上でdに**任意の1バイトの数**を指定します。すなわち

OOH～FFH

のどれでも結構です。その上でこの命令を実行すると，IYとdを加えた値の番地にデータが転送されます。たとえば，IY=0F7D8Hのとき，

〈例〉 dの値　　IY+d
00H ⟶F7D8H+00H=F7D8H
01H ⟶F7D8H+01H=F7D9H
02H ⟶F7D8H+02H=F7DAH

のそれぞれの番地にデータが転送されるわけです。

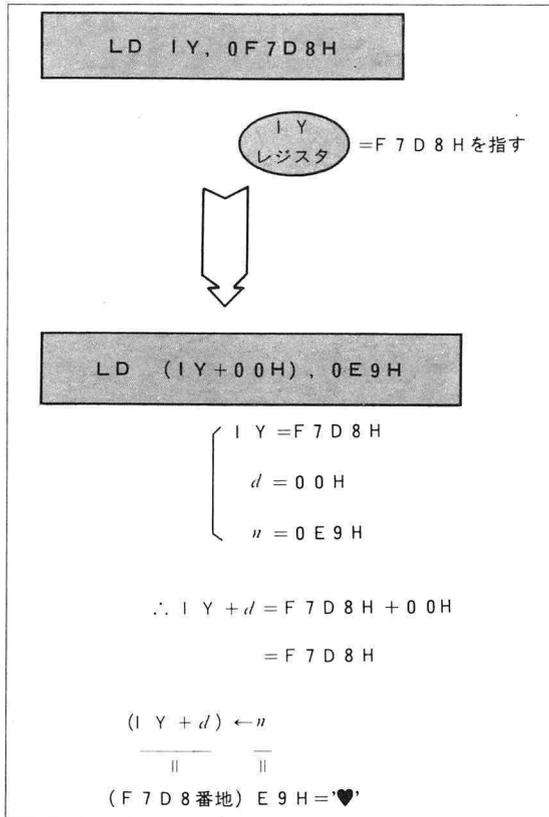
以上のことを頭に入れた上で，もう1度**第3-9図**をふりかえってみましょう。

① IYレジスタに，ビデオRAMの番地F7D8Hをセットします（1行目）。
② 2行目で，

(IY＋00H)
d＝00Hだから，F7D8番地を指す

により，♥のキャラクタ・コードE9HをF7D8 番地に移します。

③　モニタへジャンプする（3行目)。

以上の手順を図解すれば，**第3-10図**のようになります。

《第3－10図》第3－9図のプログラムの分解

最後にハンド・アセンブル時のマシン語への変換のところを見ておきます。

①　**1行目**

FD　21

の2バイトが，

LD　IY, nn

のマシン語です。そのあとに，IYにセットする2バイトのデータを上位，下位を逆にしてつけます。

セットしたい値＝F7D8H
マシン語：D8　F7

結局1行目をハンド・アセンブルすると，

FD　21　D8　F7

になります。

②　**2行目**

FD　36

の2バイトが，

LD　(IY＋d),　n

のマシン語です。そのあとにd, nの順にマシン語を書きます。結局2行目は，

FD　36　00　E9

となります。

以上で，インデックス・レジスタを用いた場合の説明をおわります。さっそく**第3-8図**，**第3-9図**のプログラムを順にあなたのマシンで走らせてみてください。**第3-7図**と同様に画面中央に♥が表示されるはずです。

11. お慰み――Mr.Xのために

前節の“**インデックス・レジスタ**”いかがだったでしょうか？　一部の方には，少し難しかったかもしれません。まあ，少なくとも少しは頭が混乱したことでしょう。

しかし，前節の内容をいますぐに完全に理解する必要はありません。なぜなら，いまは〈チャレンジ2〉の解答例をいろいろな角度から紹介し，合わせて**レジスタのいろいろな使い方**を知っていただこうとしているわけで，いわば**〈チャレンジ2〉解答例の品評会**を催しているようなものです。したがって，そのどれもこれも完全に覚える必要はなく，

どれか一つ!

あなたの気に入った方法で画面表示ができるようになれば結構です。

同様の主旨で，次節では**レジスタ・ペア**を用いた方法を紹介します。しかし，これについても今すぐに理解しなければいけないという性格のものではなく，本書の演習が進む中で，いつか必要がおきたとき，理解していただけば良いでしょう。

12. HLレジスタを用いて

本節では，〈チャレンジ2〉への第三の方法によるアプローチを試みます。そのために最初に

レジスタ・ペア

という概念を明らかに致しましょう。

最も一般的な8ビット・レジスタである

B, C, D, E, H, L

は，二つのレジスタをくっつけて

"B+C"で"BCレジスタ"

"D+E"で"DEレジスタ"

"H+L"で"HLレジスタ"

と呼び，あたかも

16ビット・レジスタ

であるかのように振舞います。これを，レジスタをペアにして使うため，**レジスタ・ペア**と呼んでいます。

たとえば，

Bレジスタ←—12H

Cレジスタ←—34H

を代入することを考えてみましょう。これまで我々の知識だけでしたら，

```
LD B, 12H  ⎫
LD C, 34H  ⎭———①
```

のようにプログラムするでしょう。ところが，B，Cレジスタをペアにし，**BCレジスタ**とすれば，

```
LD BC, 1234H
```

のように①のプログラムを，1行で書くことができます。

そこで，レジスタ・ペアによる〈チャレンジ2〉の解答例を示しましょう。三つのレジスタ・ペアのうち

HLレジスタ

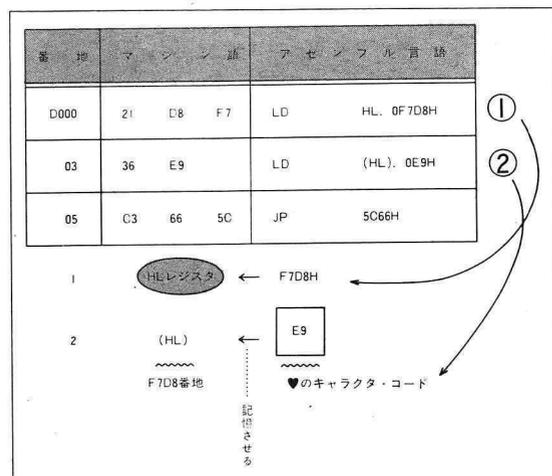
だけは特別で，あたかもインデックス・レジスタのように番地のポインタとして使うことができます。ここでは，その性質を使います。

第3-11図をみてください。これがそのプログラム例です。

番地	マシン語	アセンブル言語
D000	21 D8 F7	LD HL, 0F7D8H
03	36 E9	LD (HL), 0E9H
05	C3 66 5C	JP 5C66H

1　HLレジスタ ← F7D8H　①

2　(HL) ← E9　②

F7D8番地　　記憶させる　　♥のキャラクタ・コード

《第3-11図》HLレジスタを使って

① **1行目**

LOCATE 40, 10

に相当するビデオRAMの番地F7D8Hを，レジスタ・ペアHLにセットします。

② **2行目**

またまた( )が出てきましたね。

HLと(HL)

どう違うのでしょうね。これについては，後述します。ここでは，HLの指す番地（すなわちF7D8番地）にE9H（♥のキャラクタ・コードです）が格納されるものと思ってください。

③ **3行目**

説明不要ですね。

以上で，レジスタ・ペアによる解答例の説明を終ります。このプログラムについても実際に確めてみてください。

13.5キャラクタに挑戦

前節までで我々は

好きなキャラクタを

TV画面に好きな位置に表示

する方法を**三つ**こなしてきました。この時期には，とにかく**理屈を考えているよりは，いろいろなキャラクタを，いろいろな場所に表示してみる**ことをお勧めします。何回か実験しているうちに，マシン語のプログラムが反射的に書けるようになるでしょう。

さて，〈チャレンジ2〉では，表示の内容が**1キャラクタ**でした。これでは，実用上不便です。そこで次の演習を試みましょう。

〈チャレンジ3〉

```
LOCATE 30, 5
PRINT "◢◢◢◢◢"
```

のプログラムをマシン語で実現しなさい。

これは，"◢"のキャラクタを五つ横に並べる問題です。まずは，**自分で知っている範囲内の知識を使ってプログラムを完成**させてみてください。しかる後に，次の解答に進んでください。

まず，どんな方法をとろうとも，プログラムを組む前に準備が必要です。そうです。

ビデオRAMの番地＝レイアウト・シート

キャラクター・コード＝キャラクター・コード表

による変換作業です。これは，OKですね？　この変換作業の結果を，**第3-12図**にまとめておきました。

◢ ──→ 〈キャラクタ・コード〉＝E4H

座　標		ビデオRAM
LOCATE	30,5	F576H
LOCATE	31,5	F577H
LOCATE	32,5	F578H
LOCATE	33,5	F579H
LOCATE	34,5	F57AH

《第3-12図》準　備

番地	マシン語			アセンブリ言語		
D000	3E	E4		LD	A, 0E4H	①
02	32	76	F5	LD	(0F576H), A	②
05	32	77	F5	LD	(0F577H), A	③
08	32	78	F5	LD	(0F578H), A	④
0B	32	79	F5	LD	(0F579H), A	⑤
0E	32	7A	F5	LD	(0F57AH), A	⑥
11	C3	66	5C	JP	5C66H	⑦

《第3-13図》Aレジスタによる基本例

次は実際のプログラム例です。**第3-13図**をみてください。これはAレジスタを使ったもので，最も わかりやすい例です。読者もこの方法で組まれた方が多いと思います。すぐに理解できると思いますので，簡単に説明しておきましょう。

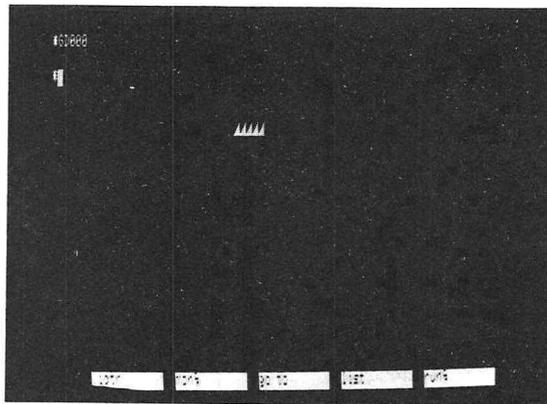
① まず1行目で，

Aレジスタ←──◢のキャラクタ・コード

のようにセットします。

② 2～6行目で，各ビデオRAMにAレジスタのコードを転送します。すなわちこの作業により，TV画面に表示されるわけです。

③ 7行目でモニタにジャンプします。

このプログラムを実行させた様子を**写真8**に示しておきます。

《写真8》プログラム実行

さて，**第3-13図**のプログラム，いかがでしたか？　2～6行目が同じこの繰り返しで，何となく幼稚ぽくスッキリしませんね。インデックス・レジスタを使えば，この辺の事情が少し改善されます。**第3-14図**をみてください。インデックス・レジスタの使い方が，前と少し変えてありますから説明しておきましょう。

番地	マシン語				アセンブリ言語		
D100	3E	E4			LD	A, 0E4H	①
02	DD	21	76	F5	LD	IX, 0F576H	②
06	DD	77	00		LD	(IX+00H), A	③
09	DD	77	01		LD	(IX+01H), A	④
0C	DD	77	02		LD	(IX+02H), A	⑤
0F	DD	77	03		LD	(IX+03H), A	⑥
12	DD	77	04		LD	(IX+04H), A	⑦
15	C3	66	5C		JP	5C66H	⑧

《第3-14図》インデックス・レジスタによる改善

① **1行目**

Aレジスタ←──◢のキャラクタ・コード

② **2行目**

IXレジスタ←──ビデオRAMの番地

③ **3～7行目**

まず3行目では，

IX＋00 H＝0 F576H

ですから，そこにAレジスタに格納されている◢のキャラクタ・コードを転送します。

(注) 一般に“LD X_1, X_2”でX_2の内容がX_1に転送されますが，**転送前後でX_2の内容は変わりません。**ですからLD命令実行後は，

X_1の内容＝X_2の内容

になります。

この注意は重要です。したがって，3行目でAの値を（IX＋00H）に転送していますが，それを実行したといってAの内容＝E4Hが変わるわけではありません。そのAの値（“◢”）を4行目で

IX＋01 H＝0 F577H

に転送しています。すると先程の右隣にも◢が表示されます。

こうして，3～7行でAレジスタに入っている◢を次々に右隣に転送し，五つの◢の列がプリントされるわけです。

④ **8行目**

OKですね？

以上がインデックス・レジスタによるプログラム例です。いかがでしたか？ **第3-13図**よりは，メモリを喰いますがスッキリしていると思います。ここで**第3-14図**のプログラムについて，すこし補足をしておきます。

① いつも同じ番地では面白くありませんから，ここではD100番地にプログラムを割り当ててみました。したがってプログラムを走らせるには，

GD100↗

となります。

② ここでは**IXレジスタ**を使いましたが，**IYレジスタ**を使っても同様に可能です。自分でプログラミングし，ハンド・アセンブルしてみてください。

③ さらにここでは，キャラクタ・コードを格納するのに**Aレジスタ**を使っていますが，他の

B, C, D, E, H, Lレジスタ

でも可能です。これについても，自分で実験してみてください。こうして自分なりに試行錯誤をくり返していれば，だんだんと“Z－80活用表”に慣れてくると思います。

さて，今はインデックス・レジスタを使いましたがHLレジスタを使えばさらにスッキリしたプログラムが作れます。しかも，**省メモリ**のです。それには，若干の予備知識が必要です。そこで節を改めて取り組んでみましょう。

14. 増減命令をマスターする

まず解答例から見ていただきましょう。**第3-15図**です。この3行目を見ると，

INC HL

と書いてあります。見慣れない命令ですね。

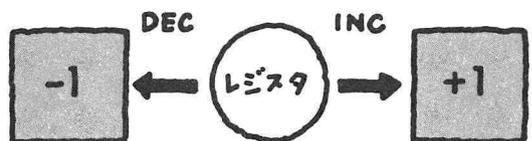
番地	マシン語			アセンブリ言語		
D000	06	E4		LD	B, 0E4H	①
02	21	76	F5	LD	HL, 0F576H	②
05	70			LD	(HL), B	③
06	23			INC	HL	④
07	70			LD	(HL), B	⑤
08	23			INC	HL	⑥
09	70			LD	(HL), B	⑦
0A	23			INC	HL	⑧
0B	70			LD	(HL), B	⑨
0C	23			INC	HL	⑩
0D	70			LD	(HL), B	⑪
0E	C3	66	5C	JP	5C66H	⑫

《第3-15図》HLレジスタによる改善

INC命令

レジスタやメモリの内容を＋1する。

DEC命令

レジスタやメモリの内容を－1する。

INC命令，DEC命令は，Z-80の演算命令の中ではもっとも基本となるもので，この二つをまとめて

増減命令

と呼んでいます。非常に簡単な命令ですが，これを知らないと今後どうしようもありませんから，ここで良く理解しておいてください。これらの命令については**第3-16図**にまとめておきました。しかし，すぐにこれらのすべてを理解するのは難しいと思いますが，以下の説明を読む際の参考にしてください。

<table>
<tr><th colspan="2">命　令</th><th colspan="4">増　減　の　対　象</th></tr>
<tr><td rowspan="4">増
加</td><td rowspan="4">INC</td><td rowspan="3">8ビット</td><td colspan="2">汎用レジスタ</td><td>A, B, C, D, E, H, L</td></tr>
<tr><td rowspan="2">メモリ</td><td>ポインタ＝HL</td><td>(HL)</td></tr>
<tr><td>ポインタ
＝インデックス・レジスタ</td><td>(IX＋d), (IY＋d)</td></tr>
<tr><td rowspan="3">16ビット</td><td colspan="2">レジスタ・ペア</td><td>BC, DE, HL</td></tr>
<tr><td rowspan="3">減
少</td><td rowspan="3">DEC</td><td colspan="2">インデックス・レジスタ</td><td>IX, IY</td></tr>
<tr><td colspan="2">専用レジスタ</td><td>SP</td></tr>
</table>

《第3-16図》増減命令のまとめ

簡単にいえば，

INCは，＋1する命令

DECは，－1する命令

という意味です。例をあげてみます。**第3-17図**をみてください。最初，

Aレジスタ＝28H

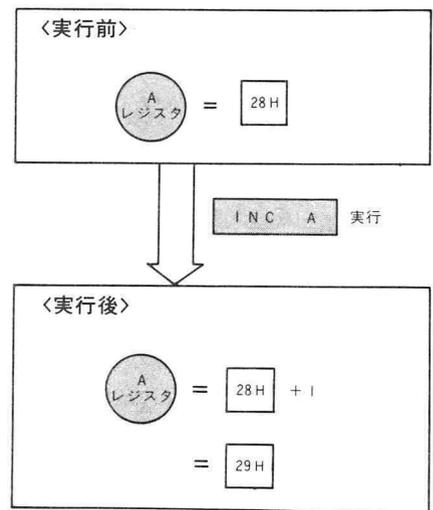
が記憶されていたとします。このとき，

INC A

という命令を実行すると，Aレジスタの記憶内容28Hが**一つ大きく**なり，

Aレジスタ＝29H

となります。

DEC命令についても，例をあげておきます。**第3-18図**をみてください。今度は，16ビットのHLレジスタをとり上げてみます。最初は

《第3-17図》INC命令

HLレジスタ=7ACBH

が記憶されていたとします。このとき,

DEC　HL

という命令を実行すると，HLレジスタの記憶内容7ACBHの値が一つ小さくなり,

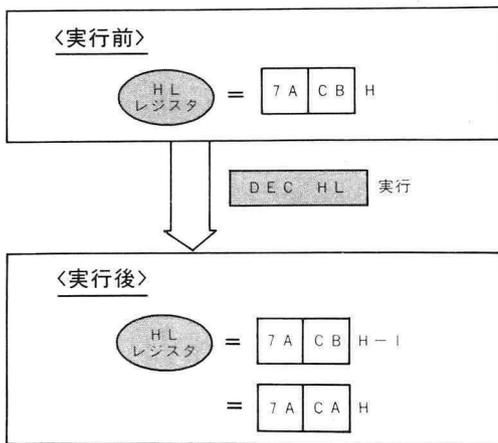
HLレジスタ=7ACAH

になります。

ちなみに,

INC=Increment

DEC=Decrement

の略です。増減命令は，他にもたくさんあります。**第3-16図**や“Z-80活用表”をときどきご覧になり，少しづつ覚えていってください。

《第3-18図》DEC命令

さあ，おおむね増減命令についてはご理解いただけたと思いますので,保留しておいた**第3-15図**のプログラムを見てみましょう。

① **1行目**

いつもAレジスタばかり使ってきましたので，ここではBレジスタを使ってみました。ここに◢のキャラクタ・コードをセットします。

② **2行目**

HLレジスタにビデオRAMの

LOCATE 40, 10

に相当する番地をセットします。

③ **3行目**

Bレジスタにセットされた◢を,HLにセットされたF576番地に送ります。すると

LOCATE 40, 10

の位置に◢が表示されます。

④ **4～5行目**

HLの値をINC命令により一つ大きくします。

HL=F576H+1

=F577H

になりますから，HLは右隣の

LOCATE 41, 10

を指します。ここに5行目で再び◢のコードを送ってやります。

⑤ 以上の操作を5回繰り返すことで，◢が五つ表示され，所期の目的を達成することができます。

HLレジスタを用いた解答例，いかがだったでしょうか？　今までの中では，スッキリ・省メモリでモア・ベターだったと思います。いちおう以上をもちまして**〈チャレンジ3〉**の解答例は，打ち止めにしておきましょう。

15. 表示数を増大させる

〈チャレンジ3〉では,五つのキャラクタの表示に挑戦しました。そこでは,

・Aレジスタによる方法

・インデックス・レジスタによる方法

・HLレジスタによる方法

の三つを紹介しました。しかし，いずれの方法でも基本的には,

“一つのキャラクタを表示する”

を5回繰り返す

というものでした。〈チャレンジ3〉では，表示回数がわずか5回でしたからこれで済みましたが，**もっとたくさんのキャラクタを表示しようとする**と，この方法を踏襲するかぎり，非現実的になってきます。たとえば次のようなプログラムを作ろうとしたら，あなたはどうしますか？

> **〈チャレンジ4〉**
>
> TV画面の第4行目を,すべて“■”で埋めるプログラムを作りなさい。

簡単です。

```
10  LOCATE  0, 3
20  FOR  I = 1  TO  80
30      PRINT  "■";
```

40　NEXT

これではお話しになりませんね。

この問題を我々の知識の範囲内でマシン語のプログラム化しようとすると，かなり**動物的労力**を必要とします。なぜなら最も合理的なＨＬレジスタを用いるプログラムでも，１キャラクタを表示するのに最低

LD　(HL)，B〔B＝■〕

INC　HL

の2命令が必要ですからこれを80回繰り返すとなると，

何と160行以上のプログラム！

になってしまいます。画面の１行を“■”で埋めるだけのためにこんなことをやっていたのでは，オール・マシン語で“スペース・インベーダー”でも組んだら**曾おじいさん**になってしまいます！

先にBASICによるプログラム例をお目にかけました。何でBASICだと，あんなに簡単に書けてしまうのでしょう？　理由は簡単です。

BASICは，

FOR～NEXT

という繰り返し命令を持っている

からです。それでは，マシン語では？　あります，あります。

マシン語にも繰り返し処理専門

の命令が存在

します。次節では，その命令に焦点を当ててみましょう。

16. DJNZの基本

それは，

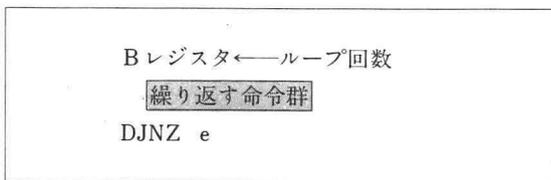
DJNZ

という命令です。

> **DJNZ命令**
>
> **マシン語**：10　e—2
>
> 〔e：レラティブ・アドレッシング・モードにおける変位置〕
>
> **機能**：繰り返し処理をおこなう。繰り返し回数は，Ｂレジスタにセットする。

このDJNZという命令は，Z－80特有の命令で，その前身である8080CPUにはありませんでした。Z－80になって追加されたものです。Z－80の命令解説書では，DJNZは普通**おしまいの方で解説**されています。というのは，この命令を厳密に解説しようとすると，かなりの予備知識が要求されるからです。たとえば，**“レラティブ・アドレッシング”**にしても，いきなりこれを解説するのは無理で，やはりアドレッシング・モードの最初のから学んでいただく必要があります。しかし，その**機能に着目**すれば，**DJNZ**を理解することはそれほど難しくありません。それでは，以下を注意深くお読みください。

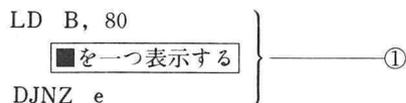
まず，命令の使い方は次のようになります。

> Ｂレジスタ←―ループ回数
>
> 繰り返す命令群
>
> DJNZ　e

たとえば，〈チャレンジ４〉の場合でしたら，

ループ回数＝80回

繰り返す内容＝■を１つ表示する

ですから，プログラムの概略は次のようになるでしょう。

```
LD　B，80          ┐
  ■を一つ表示する   ├――――①
DJNZ　e            ┘
```

後はこのプログラムに肉づけしていけばよいのです。なお３行目の“e”については，のちほど説明します。

ところで①の中には，１ヶ所

間違い

があります。おわかりでしょうか？

そうです。１行目です。１行目でＢレジスタにループ回数を代入していますね。ところが，先に

８ビット・レジスタには，

8ビット＝2桁の16進数しか代入できないと説明しました。Ｂレジスタは，もちろん８ビットのレジスタです。このため80をそのままＢレジスタに代入することはできません。

80を2桁の16進数に変換

してやる必要があるのです。

ところで，あなたは，

80という10進数を16進に変換

できますか？　もちろん，本書では説明してありません。先に何度も述べましたが，**こんなことは，コンピ**

ュータをやる上で常識中の常識ですから，いやしくもマシン語をやろうとするあなたは，どこかで身につけているはずです。

それはさておき，80を16進数に変換する最も簡単な方法は

80＝50H

と覚えてしまうことです。これはビックリすることでも何でもなく，マシン語をちょっとでもやっていれば良く出てくる数は自然と覚えてしまいます。

次に簡単な方法は，付録の**“10進←→16進変換表”**を利用することです。まあ，いずれにしても

80＝50H

と変換されますから，①は次のように訂正されます。

```
LD   B, 50H
     ■を１つ表示する     ──── ②
DJNZ e
```

以上で正しいプログラムの骨格はできあがりました。その肉付けを次節で考えていくことにします。

（注）**“アセンブラ”**（アセンブル作業を自動的にやってくれる）というプログラムを使うと，①のような形でプログラムを書いておいても，アセンブル時に**10進数を自動的に16進数に変換**してくれます。しかし，人間がアセンブル作業を行うときは**ミスを事前に防ぐ**ために，アセンブリ言語の段階で②のような16進数で書いておく方がよいでしょう。

17. リピートの内容は？

そこで②の肉付けを行います。

繰り返し処理の中身を考えましょう。

“■を一つ表示する”

わけです。それには，いろいろな方法がありました。ここではHLを使ってみましょう。

Aレジスタ←──■のキャラクタ・コード

HLレジスタ←──ビデオRAMの番地

を入れておき，

LD (HL), A

といれればいいですね。そこで②に表示の部分を追加すると次のようになります。

```
LD   B, 50H
LD   (HL), A    ──── ③
DJNZ e
```

ただし，このままではA, HLレジスタの設定がされていませんから，何が起こるかわかりません。そこでデータの準備をします。

■のキャラクタ・コード＝87H

これは，すぐわかりますね？　ビデオRAMの位置はどこを指定してやればいいのでしょうか？　**〈チャレンジ４〉**では，“第４行目に”と書いてあります。したがって第４行目の左端の位置，LOCATE座標でいえば，

LOCATE　0，3

を指定してやればいいでしょう。したがって

ビデオRAMの番地＝F468H

となります。

データの準備ができましたので，A, HLの各レジスタをその値に設定してやります。

```
LD    A, 87H
LD    HL, 0F468H
LD    B, 50H       ──── ④
LD    (HL), A
DJNZ  e
```

さて，これでいいでしょうか？　よく④のプログラムを目で追ってみてください。何かが不足していませんか？　④では，

LD (HL), A

の部分が80回繰り返されます。ところが**HLはつねに**

LOCATE　0, 3

を指しているため，80回ともそこに表示されてしまいます。これでは，TV画面の第４行目左端にしか■が表示されませんね。

４行目をすべて■で埋めてやるには，■を一つ表示するたびにHLの値を一つ右に移してやる必要があります。つまり，HLの値を**一つ大きく**してやればいいのです。それには，増減命令でやった

INC HL

を使えばいいですね。ついでにプログラムの最後に

JP　5C66H

を追加しておきましょう。

```
LD    A, 87H
LD    HL, 0F468H
LD    B, 50H
LD    (HL), A      ──── ⑤
INC   HL
DJNZ  e
JP    5C66H
```

さあ，これでプログラムの主要な部分は終りました。そろそろ“e”の意味に進むときがきたようです。次節，頑張りましょう。

18.“e”の値——DJNZのしくみ

まず“e”の部分は気にしないでハンド・アセンブルしていきます。

DJNZ　e

のマシン語は，2バイトで

1 0 ××
↑
(この部分はあとで説明する)

となりますから，不明の部分はとりあえず一でも引いておきます。ハンド・アセンブルが完了したら，番地を割り当てます。例によってD000番地から割り当てることにします。各行に番地を割りふるときは，一の部分も抜かさないように1バイトとして数えます。ここまでの作業を終えたプログラムが**第3-19図**です。

番地	マシン語	アセンブリ言語	
D000	3E 87	LD A, 87H	①
02	21 68 F4	LD HL, 0F468H	②
05	06 50	LD B, 50H	③
07	77	LD (HL), A	④
08	23	INC HL	⑤
09	10 ——	DJNZ ————	⑥
0B	C3 66 5C	JP 5C66H	⑦

《第3-19図》'e'を無視してハンド・アセンブルする

ここまでくれば，“e”の値はすぐにわかります。

e＝繰り返し処理のスタート番地

このように考えてください。**第3-19図**のプログラムでは，

LD (HL), A

が繰り返し処理の最初になりますから，4行目，すなわち

D007番地＝e

ということになります。この“e”の値をアセンブリ言語の6行目に書き込むと，**第3-20図**のようになります。

番地	マシン語	アセンブリ言語	
D000	3E 87	LD A, 87H	①
02	21 68 F4	LD HL, 0F468H	②
05	06 50	LD B, 50H	③
07	77	LD (HL), A	④
08	23	INC HL	⑤
09	10 ——	DJNZ 0D007H	⑥
0B	C3 66 5C	JP 5C66H	⑦

《第3-20図》'e'の値を追加する

これで“e”の値は，わかりました。それを含めてDJNZの使い方をまとめると，次のようになります。

こうしてまとめてみると，おおむね

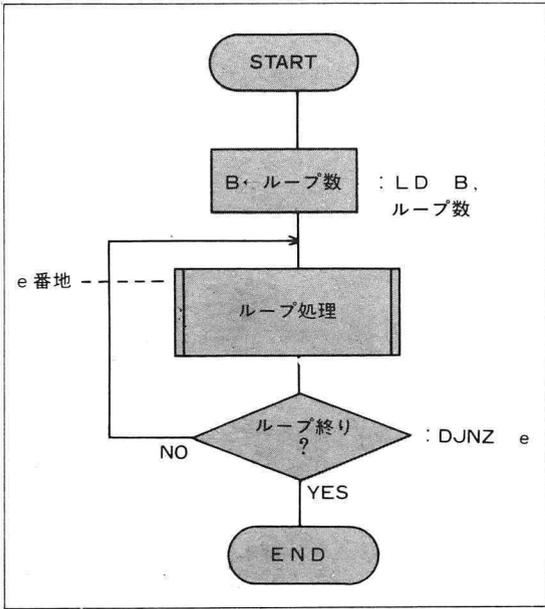
DJNZの働き(しくみ)

がわかってきたのではないでしょうか？　そうです。そのしくみは次のようになっています。

①Bレジスタにループ数をセットする。

②1回分の処理を行う。

③判定——必要なだけ繰り返し処理が行われたかをチェックし，そうでなければ②にジャンプする。終了したら次の命令に進む(**第3-21図**)。

さあ，残りは何でしょう？　6行目のハンド・アセンブルの部分**1バイト分**だけです。これは，どのように変換すれば良いのでしょうか？“少し変だ”——と思われた方が多いかもしれません。その理由は，6行

《第3-21図》DJNZのしくみ

目は,

DJNZ 0D007H

となっています。これをマシン語に変換するには,

```
  10  07 D0
  ↑      ↑
DJNZ e   D007H
```

とするのが自然のような気がします。ところが**第3-20図**のマシン語のところを見ると，**2バイト分しか**用意されていません。

1バイト不足では？

こう思われてきますね。しかし，もちろん**第3-20図**はあっています。それならどのようにマシン語に変換するのでしょうか？

19.「番地」の相対性理論

いま，我々は**ある番地をマシン語**で示そうとしています。**番地を直接指定しようとすると，2バイト必要**です。なぜなら，各番地は2バイトの数で構成されているからです。ところが，やり方によっては

番地を1バイトで示す

ことも可能です。

かつて19世紀にまとめられた「**ニュートン力学**」は，20世紀に入り**アインシュタインによる「相対性理論」**によりその根本が崩れ去りました。「ニュートン力学」は，ある条件のもとで近似値を示しているにすぎなかったわけです。そして「相対性理論」では，

“すべての物理法測は，互いに等速直線運動をしているすべての観測者に対して同一の形をとる。”

という「**特殊相対性原理**」とその10年後に発表された「**一般相対性原理**」により展開されてきたわけです。

たかが“番地”のことなのに，話しが大げさになってしまいました。しかし，我々もアインシュタインよろしく，番地を

絶対的に固定されたもの（**絶対番地**）

としてとらえるのではなく，もっと柔軟的に

相対的なもの（**相対番地**）

として表現してみることにしましょう。

その考え方は，次のようなものです。

郵便屋さんは郵便物を絶対番地で配達します。たとえば,“品川区東五反田1—11—15」と書いてあるものは「電波新聞社」に配達されます。ところで，五反田の街角で郵便屋さんに道を聞いたとします。すると，

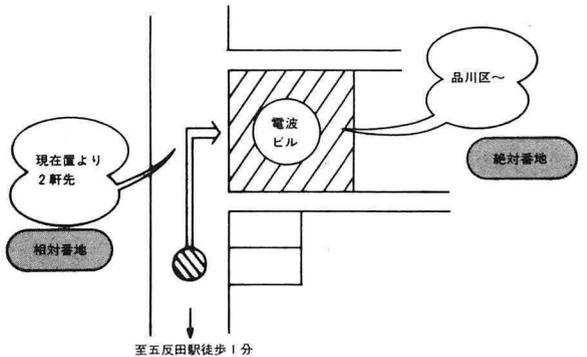
「2軒先の大きなビルですよ」

と教えてくれるかもしれません。それでもやはり「電波新聞社」に到達しますね。

このようにある位置を示すには，

絶対的な住所（**絶対番地**）で示す。

現在地からの相対的な位置(**相対番地**)で示す。

の二通りの方法があるわけです（**第3-22図**）。

《第3-22図》「電波新聞社」は？

20. プログラム・カウンタ

この考え方を，マシン語の番地にも応用します。

ある基準を考え，そこから10バイト手前の番地とか125バイト先の番地とかのように相対的な番地を考えるのです。この番地を

相対番地

と呼んでいます。

そこで番地を相対的に示すわけですが，このとき

どこを基準にするか？

が問題です。たとえば，単に

「５バイト手前の番地」

といったのでは，どこを指しているのかわかりません。しかし，きちんと基準を示し，

「D010番地の５バイト手前」

といえば，はっきりD005番地とわかります。

相対番地の基準を示すには，一つ予備知識が必要です。我々は，第２ブロック・第３章においてレジスタの一覧表を見ました。その中の“**専用レジスタ**”の一つに16ビットの

PC（プログラム・カウンタ）

というのがあったのですが，覚えておられるでしょうか？

> **PC＝プログラム・カウンタ**
> **(program counter)**
> 次に実行するプログラムの番地を記憶しているアドレス。プログラムの実行にともない自動的にインクリメントされる。

Z―80CPUでは，PCという16ビット・レジスタを持っていて，次に実行すべき番地を記憶しています。**第3-20図**のプログラムを見てください。このプログラムが，メモリに入っているとします。このプログラムを実行しようとすると，

GD000↗

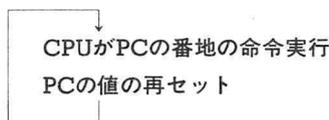
とキーインしますね。これを実行するとモニタは，

PC←―D000H

をセットします。CPUは，このPCの値を見てD000番地に書かれている命令を実行します。このときＰＣの値は自動的に

PC＝D002H

にセットされなおされています。そこでまたPCの値を見て，D002番地の命令を実行していくわけです。すなわち，

CPUがPCの番地の命令実行
PCの値の再セット

という具合にプログラムが進行していくことになります。

21. 符号付16進数

もうおわかりだと思いますが，

相対番地では，PC(プログラム・カウンタ）の指す番地を基準におく。

のです。もう一度**第3-20図**をみてください。今やりたいことは，**６行目から４行目へ戻したい**ということです。６行目の命令の実行が終ったとき，PCは７行目の先頭を指していますから，

PC＝D00BH

になっています。ここを基準に数えると４行目のD007番地は何バイト手前でしょうか？ 命令のバイト数を数えてください。

４バイト手前

であることがわかります（**第3-23図**)。すなわち

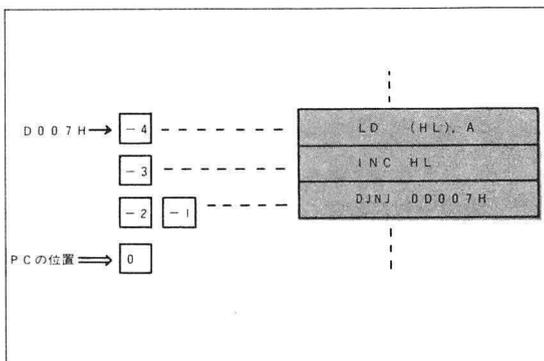
D007H＝PC―4

というわけです。この

―4

がD007番地の相対番地ということになります。したがって６行目をハンド・アセンブルすると

10　―4

となります。

《第３-23図》PCより－４バイト手前

オヤ？　おかしいですか？

「マシン語は，**16進数2桁で構成される**のだから，マシン語に—4を使うのはおかしい！」

正にそのとおりです。ところで

　　—4＝FCH

なのはご存知でしょうか？

16進数は，コンピュータをやる上で常識中の常識となっているのですが，

　　符号付き16進数

まで知っている人は少ないようです。しかし，これを知らなくてはマシン語には手も足も出ませんから，知っておく必要がおおいにあります。たとえば，1バイトでマイナスの数を表現しようとすると，次のようになります。

　　－1＝FFH
　　－2＝FEH
　　－3＝FDH
　　－4＝FCH
　　－5＝FBH
　　　⋮

この**符号付き16進数**については，感覚的に理解していただけるよう，付録に**「1バイトの符号付き16進数表」**をつけておきました。必要な時にこの表を見て変換をおこなうといいでしょう。

以上で全てがおわかりになったと思います。結局**第3-20図**の6行目は，

　　10 FC

とハンド・アセンブルされます。こうして完成したプログラムが**第3-24図**です。

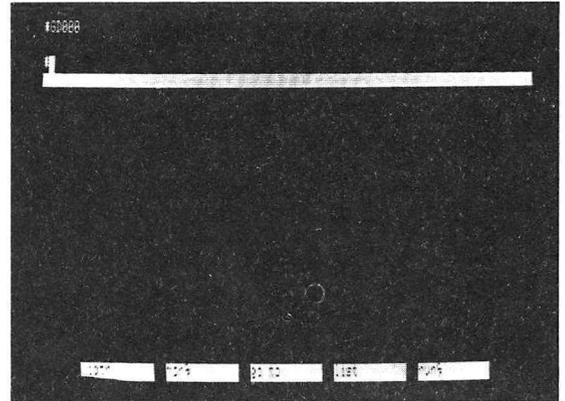
さあ，長らくかかりましたが，ここにようやく**〈チャレンジ4〉**の解答ができあがりました。さっそく実行してみましょう。**写真9**にその様子を示します。ここで注目していただきたいのは次の2点です。

① 80キャラクタを表示しているにもかかわらず，〈チャレンジ3〉よりも短いプログラムとなっている。これは，ひとえに**DJNZというループ命令を使ったお陰**です。

② **マシン語による高速性**

もう1度**第3-25図**にBACICによる解を示しておきます。実行結果を比べてみてください。BASICでは，左から右に向かって■がのびていくのがハッキリとわかります。しかし，マシン語では，瞬間的に

番地	マシン語	アセンブリ言語
D000	3E 87	LD A, 87H
02	21 68 F4	LD HL, 0F468H
05	06 50	LD B, 50H
07	77	LD (HL), A
08	23	INC HL
09	10 FC	DJNZ 0D007H
0B	C3 66 5C	JP 5C66H

《第3-24図》〈チャレンジ4〉完成

《写真9》■がのびていく様子

```
10 LOCATE 0, 3
20 FOR I=1 TO 80
30 PRINT "■";
40 NEXT
```

《第3-25図》BASIC版解答

1行分が表示されます。

以上，2点——**画期的なこと**だと思いませんか？そしてもう1点確認すれば，既に**あなたはそのプログラミング能力を身につけた**ことになります。

〈第2章のおわりに〉

最後のプログラム，いかがでしたか？　マシン語の高速性にびっくりしたことでしょう。ただし，これはまだ画面1行分でのできごとです。これが**画面全体に及んだとき**，さらにその高速性に驚くことと思います。次章では，その画面全体に目を向けることに致しましょう。

第3章

そして全てが消えた

〈はじめに〉

本章には，演習がたった一つしか登場しません。なぜならその演習を解くためにたくさんの予備知識が必要になるからです。

論理演算によるビット操作

スタック命令

2バイトの加算

等かなり高級な知識が，次々に登場してきます。するとその演習，かなりシビアな問題でしょうか？――それが，実は**なさけない程たいしたことではないの**です。その気になればキー一つで済んでしまうことなのです。

こんなこと，そうです。こんなことに手間ヒマかけるところにマシン語の面白さがあるのでしょうね。それにしてもさすがはマシン語，**BASICでは絶対にできないことがこの章の最後に出てきます**。お楽しみに。

22. 画面消去のアルゴリズム

それでは，我々の目を一挙に画面全体に向けることにします。

〈チャレンジ5〉

画面をクリアするプログラムをマシン語で組みなさい。

「何？　画面クリア？　くだらない！　**CLRキーを押せば良いではないか！**」

こう言ってしまってはおしまいです。BASICでプログラムを作るときでも，画面を見やすくするために最初の方で画面クリアの命令を入れますね？

PRINT CHR$ (12) ――①

こうやるはずです。

それではもし①**の命令を使わない**としたら，あなたはどうやってプログラムしますか？

たぶんLINE文を使うでしょうね。しかし今は，〈チャレンジ5〉の問題を解くことを考えています。できるだけ基本的な命令だけを使って考えてみてください。

となると，

1キャラクタ分の消去
‖
PRINT　“　”；

を1画面分繰り返せば良いのがわかります。プログラムの方針としては，**第3-26図**のようになるでしょう。すなわち，ここでは画面が

《第3-26図》 1画面消去のアルゴリズム

80×25行モード

になっているとして，

“1文字消去”を80回ループ

して**1行分の消去**とし，

“1行分の消去”を25回ループ

して**1画面分を消去**するという段取りです。

このアルゴリズムで，BASICで組んだものが，**第3-27図**です。実際にプログラムを走らせて，画面が消去されるのを確認してください。

さあ，これで〈チャレンジ5〉を解くメドがついた

```
10 LOCATE  0, 0
20 FOR  Y=1  TO 25
30 FOR  X=1  TO 80
40 PRINT  " ";
50 NEXT  X, Y
```

㊟1. プログラムを走らせる前に画面モードを80×25 にセットしておくこと。
2. あとでマシン語のプログラムと対応させるため、わざと2重ループにしてあるが、2000回ループさせれば、1重ループでも可能。

《第3-27図》BASICによる消去

と思います。そうです。マシン語でも**第3-26図**のアルゴリズムを採用すれば良いのです。我々は，ループの作り方を覚えましたね。ですからそれを使って組めば良いのです。ところが，——ここに**落し穴**がありまして，そう簡単にはいかないのです。

23.“空白”のキャラクタ・コード

マシン語でループ（繰り返し）処理を行うには

DJNZ

という命令を使いました。このとき，

Bレジスタ←—ループ回数（16進数）

を指定するのでしたね。そこで何はともあれ，ループ回数の16進数を準備しましょう。

25回——→19H

80回——→50H

これは，付録の“**10進——→16進変換表**”を利用してください。

次に処理の中心となる

1キャラクタ分の消去

を考えてみます。それには，

Aレジスタ←“空白”のキャラクタ・コード

HLレジスタ←ビデオRAMのアドレス

をセットしておき，

LD (HL), A

で消去できますね。

そこでこれらのデータも用意しておきましょう。

まず，ビデオRAMのアドレス，これは画面の左上から順に消していくとして，

HL←—F300H

をセットしておけば良いでしょう。

次に

“空白”のキャラクタ・コード

ですが，これはいくつになるでしょうか？ “キャラクタ・コード表”をみてください。良く見ると，表のアチコチに空白がちらばっています。このうちどれを採用すればいいのですか？

結論を言えば，**どれを採用してもかまいません**。ただし，“スペース・インベーダー”のようなリアル・タイム・ゲームを作るときは，

“空白”のキャラクタ・コード

を統一

しておく必要があります。やがて我々は，“スペース・インベーダー”に挑戦することになりますが，その時，この意味が良く理解していただけると思います。とにかく，“空白”のキャラクタ・コードとしてどれを採用してもかまいませんが，同一プログラム内でいろいろなコードを使うと，あとで困ることが起きますよとだけ注意しておきましょう。

ちなみにBASICインタプリタでは，スペースのキャラクタ・コードに

20H

を使っています。これは，

ASCII（米国情報交換用コード）

(American national Standard Code for Information Interchange)

に準拠しているためです。

またマシン語では普通“空白”のコードに

00H

を使います。というのは，マシン語にとって00Hという数は非常に扱いやすいのです。それについては，次節で述べます。いずれにしても，〈チャレンジ5〉のプログラムには，

Aレジスタ←—00H

を使うことにしましょう。

24. 古き時代のハイ・テクニック

本節は，いわゆる**“マイコン時代”**のなごりで，当時の涙ぐましい努力の跡をお見せします。そこに登場するのは，

省メモリのためのテクニック

です。当時のワンボード・マイコンでは，RAMが

256バイト

とか，とにかく少ないものでした。このためプログラムのわかりやすさよりは，**1バイトでも少なく組む努力**がなされました。そこには，**びっくりするようなハイ・テクニック（？）**があちこちに展開されたものです。今日のようにRAMがふんだんに使える時代にはそれらのテクニックは色褪せた感がします。

以下に紹介するテクニックは，そんな中でも有名なものです。今でも使っている人が多いし，雑誌等に紹介されるプログラムにも頻出します。したがって今すぐというわけではありませんが，知っておくと便利でしょう。ちなみに同様のテクニックは，大型機のアセンブラでも使われています。

以上のようなわけですから，先を忙ぐ人は，軽く読み流してかまいません。しかし，あとで何となくわからないことが出てきたら，もしかしたら本節以下に述べられていることかもしれません。その時は読み返してみてください。

さて，Aレジスタに“空白”のキャラクタ・コードをセットすることを考えます。

LD　A，00H――――――――①

これは，間違いありません。①をハンド・アセンブルしてみてください。

3E　00

の**2バイト**になりますね。

ところでマシン語に慣れた人なら，①のような組み方は致しません。たぶん次のいずれかを選択することでしょう。

SUB　A――――――――②

XOR　A――――――――③

①，②，③は，いずれも同じ機能を果します。否，むしろ②，③の方がたくさんの機能をしています。まあ，

Aレジスタ←――00H

という点では**三者同一**と言えます。

ところで②，③をハンド・アセンブルすると，それぞれ

②――――→97

③――――→AF

で**1バイト**で済みます。何と①の半分で良いのです。

さて以上の2つのうち，②についてはSUB命令が出てきたところで自分で考えてみてください。ヒントは，

Aレジスタ－Aレジスタ＝0

で当り前です。③については，論理演算の練習を兼ねて次節で説明致します。

25. 論理演算を使って

読者の皆さんは**論理演算**はご存知ですね？

AND

OR

等で，学校の数学でやったことがあると思います。ところで論理算は知っているが，プログラムの中でそれをどのように使うかは，案外知られていないようです。事実，**N―BASICでも論理演算の命令をもっています**が，雑誌等に発表されたプログラムを見ても，これらの命令を使ったものはあまり見かけられません。

そこで**論理演算そのもの**の説明は致しませんが，その**命令の使い方**を説明しておきましょう。

Z―80の命令のうち論理演算できるのは，

AND：論理積
OR：論理和
XOR：排他的論理和(exclusive or)
CPL：否定(反転＝complement)

の四つです。いちおうこれらの演算は既知とします。その真理値表を**第3-28図**に掲げておきます。

これらのうち，**XOR（排他的論理和）**を例に取り上げることにします。そこで**第3-29図**をみてください。2行目にXOR命令が出ています。その2行目以外は簡単にわかりますね。

①　**1行目**

Aレジスタに63Hをセットします。

①AND

0	0	0
0	1	0
1	0	0
1	1	1

②OR

0	0	0
0	1	1
1	0	1
1	1	1

③XOR

0	0	0
0	1	1
1	0	1
1	1	0

④CPL

0	1
1	0

《第3-28図》真理値表

D800	3E	63		LD	A, 63H
02	EE	D7		XOR	0D7H
04	32	00	E0	LD	(0E000H), A
07	C3	66	5C	JP	5C66H

《第3-29図》XORを使って

② **2行目**

何かやっています。

③ **3行目**

Aレジスタの値をE000番地にストアしています。プログラム終了後, E000番地の内容を見ることにより, Aレジスタの値を確認できます。

④ モニタへジャンプします。

さあ, そこで2行目の意味は後回しにして, このプログラムを走らせてみることにします。キーインし,

GD800↗

何も起こりません! このあと何をしたらいいのでしょうか? そうです。Aレジスタの値を確認するため

SE000↗

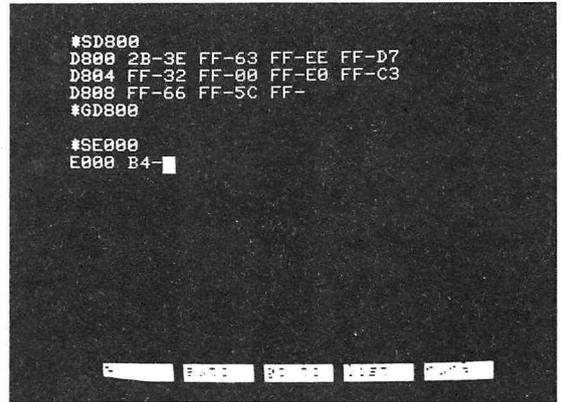
をやればいいですね(**写真10**)。プログラム終了時に

Aレジスタ=B4H

になっているのがわかります。

第3-29図のプログラムにおいて実行時に

Aレジスタの値

スタート時:63H
終了時:B4H

《写真10》Aレジスタの値を確認

のように変化したのがわかります。それでは, 一体どの時点でAレジスタの値が変化したのでしょうか?

26. XORの実際

それは, もちろん2行目の

XOR命令

が原因です。

> **XOR命令**
>
> **〈書式〉**:XOR S
>
> **〈機能〉**:AレジスタとSとの排他的論理和をとり, Aレジスタに格納する
>
> S:A, B, C, D, H, L
> (HL)
> (IX+d), (IY+d)
> n(8ビットの数)

それでは, 具体的に説明致します。

XOR命令は, Aレジスタの内容と何かの排他的論理和を計算し, その結果をAレジスタにしまう命令です。**第3-29図**の2行目でしたら

Aレジスタの内容:63H
何か:D7H

ですから

(63H) ⊻ (D7H)――――①

の論理計算をするわけです。**⊻は, 排他的論理和を表わす記号**です。

次に①の計算をどのようにするのかを示します。

まず論理演算を行う二つの16進数を，２進数に変換します。この場合でしたら，

63H⟶0110　0011

D7H⟶1101　0111

と変換されます（付録の“**16進数⟷2進数変換表**”を利用してください）。

こうしてできた８ビットの数を，**ビット毎に排他的論理和**をとります。まず１番右側のビットを見てみましょう。どちらも１ですから

$1 \veebar 1 = 0$

により結果は０です（この計算は，**第3-28図**の真理値表を見れば良いのですよ）。次のビットも同様に０になります。その左は，０と１ですから

$0 \veebar 1 = 1$

です。こうして次々にビット毎の論理演算を行っていくと，次のようになります。

```
63H ⟶ 0 1 1 0   0 0 1 1
      ↑ ↑ ↑ ↑   ↑ ↑ ↑ ↑
        [ X O R ]
      ↓ ↓ ↓ ↓   ↓ ↓ ↓ ↓
D 7H ⟶ 1 1 0 1   0 1 1 1
             ‖
      1 0 1 1   0 1 0 0
```

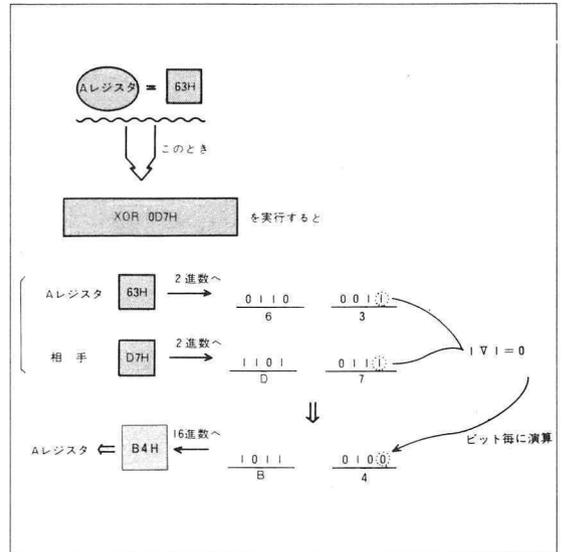
こうして，

1 0 1 1　0 1 0 0

という２進数が得られます。この２進数を16進数に逆変換すると，

B4H⟵1 0 1 1　0 1 0 0

となります。このB4Hが，何をかくそう**写真10**で見たB4Hです。

以上，排他的論理和のプログラム内での動きを見てみました。この手順を**第3-30図**にまとめておきましたので参考にしてください。他の論理演算についても同様にできますので，自分でいろいろな例を作って実験してみてください。

《第３-30図》論理演算の仕方

27. “XOR　A”を探る

以上，プログラムの中における論理演算の動きを，XORを例にして見てみました。論理演算の説明の打ち切りとして，

XOR　A

が，

LD　A，00H

の代わりになるわけを考えてみましょう。

それには，**第3-28図**③の“XORの真理値表”をじっと見てください。

同種のもののXOR⟶0

異種のもののXOR⟶1

であることがわかります。たとえば，

$1 \veebar 1 = 0$

$0 \veebar 0 = 0$

のように，**同じもののXORは0**になります。

ところで，

XOR　A

という命令は，

AレジスタとAレジスタ

の排他的論理和をとる命令ですから，

同じものどうしのXOR

ということになり，**演算の結果は０**になります。論理演算命令では結果がAレジスタに格納されるのでした

ね。したがって，

　　Aレジスタ←―00H

ということになり，

　　LD　A，00H

と同じ結果になります。

これまでは抽象的に説明しました。一つ例を挙げてみましょう。たとえば，

　　Aレジスタ←―5EH

だったとします。このとき，

　　XOR　A

を実行すると，

```
5EH───→0 1 0 1   1 1 1 0
       ↑ ↑ ↑ ↑   ↑ ↑ ↑ ↑
            [XOR]
       ↓ ↓ ↓ ↓   ↓ ↓ ↓ ↓
5EH───→0 1 0 1   1 1 1 0
                ||
0 0 H←──0 0 0 0   0 0 0 0
```

となり，所期の目的が達成されたことになります。

28. 2重ループの悲劇

前節で〈論理演算〉についての扱い方はおわかりいただけたと思いますので，〈チャレンジ５〉の解答に戻ります。

現在，我々が合意に達しているのは，次の２点です。

・**最初に消去するビデオRAMの位置＝F300**

・**“空白”のキャラクタ・コード＝00H**

これらをプログラムの最初に定義するとして，その命令は，

```
LD    HL, 0F300H
XOR   A
```

となります。また，１文字分の消去は，

　　LD　(HL)，A ─────①

となります。

次にプログラム全体の構造は，**第3-26図**にしたがうこととします。すなわち，

　　①を80回ループして**“１行分の処理”**──②

とし，

　　②を25回ループして**“画面全体の処理”**

にするわけです。これは，いわゆる繰り返し処理を２重に行う

　　２重ループ

の構造を成します。前の“画面消去のアルゴリズム”のところで予測した**“落し穴”**は，実はここに存在するのです。

それには，実際に組んでみればわかります。中側の１行処理の部分は，次のようになるのはおわかりですね？

```
        LD     B, 50H
e2番地：LD     (HL), A
        INC    HL
        DJNZ   e2
```

50Hが80の16進数表現であることは，先に見ました。この処理を25回（25＝19H）ループすれば良いのですから，消去の処理全体では，

```
e1番地：LD     B, 19H   ……………①
e2番地：LD     B, 50H   ……………②
        LD     (HL), A  ……………③
        INC    HL       ……………④
        DJNZ   e2       ……………⑤
        DJNZ   e1       ……………⑥
```

となります。これがBASIC版の解答である**第3-27図**のプログラムの20行から50行にあたります。ここでHLレジスタは，ビデオRAMの番地を示すポインタに使っていますが，ここではあまりそのことは気にしないで，ループの構造だけに注目してください。そうしますと，以上のプログラムはBASIC版同様，論理的に

　　まったく正しいプログラム

であると言えます。しかし，何だか少し変ですね。

ループの中にループを作る──この構造は理論的には正しいのです。上記のプログラムにおいて，１行目は６行目に対応し，２行目は５行目に対応してループを作っています。しかし，１行目と２行目を見るとやはりおかしいことに気づきます。なぜなら１行目で

　　Bレジスタ←―外側のループ回数

をセットしても，２行目で

　　Bレジスタ←―中側のループ回数

に直されてしまいます。これではおかしな動作をしてしまいますね。この悲劇は我々が，

　　Bレジスタを使うループしか知らない

ために起きています。**第3-27図**のBASICのプログラムを見ても，中側と外側で

　　異なる変数

を使っていますね。

29. PUSHとPOP

この悲劇を喜劇に変えるには，二つの方法が考えられます。

① **Bレジスタ以外のループの作り方を覚える。**

もちろんDJNZは使えませんが他の方法があります。

② **Bレジスタのままで対策を考える**

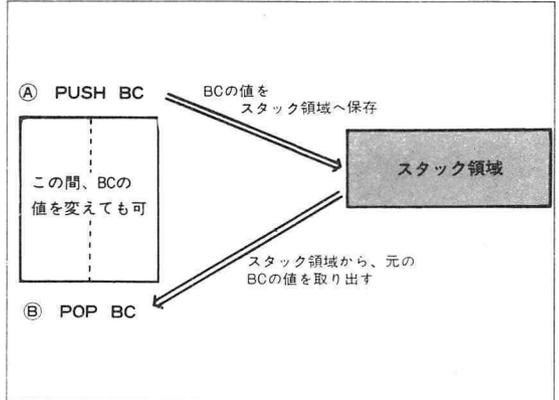
そんなことができるのか？——と思われるかもしれません。でも次のテクニックを使えば，それが可能なのです。

> **PUSH命令**
> **〈書式〉**：PUSH　AA
> AA＝AFレジスタ
> BCレジスタ
> DEレジスタ
> HLレジスタ
> IXレジスタ
> IYレジスタ
> **〈機能〉**：レジスタの値をスタック領域に待避させる。
>
> **POP命令**
> **〈書式〉**：POP　AA
> **〈機能〉**：退避させておいたレジスタの値をスタック領域から取り出す。

前に私は，“レジスタはBASICにおける変数のようなものだ”と述べました。その時
「マシン語の変数って少ないな。これだけで足りるのかな」
と思われたかもしれません。もちろん，これだけでは足りません。実は，この

PUSH命令
POP命令

があるお陰でレジスタの数が少なくて済むのです。

それでは，そのPUSH命令，POP命令を**第3-31図**を見ながら，説明することにします。いまⒶ地点でPUSH命令を実行したとします。するとその時のBCの値が**スタック領域**（メモリ上にあります）に保存されます。続いていろいろな処理をするとします。その間BCレジスタの値を変えてもかまいません。そして，BCの値を元に戻したくなった時，ⒷのようにPOP命令を実行します。するとスタック領域にしまわれていた元の値がBCレジスタに戻されます。したがって**Ⓐ時点とⒷ時点でBCレジスタの値は同じ**ということができます。

《第3-31図》PUSH，POP命令

30. PUSHとPOPの実験

さて，以上のことを実験により検証してみたいと思います。

第3-32図のプログラムをみてください。

番地	マシン語				アセンブリ言語		
D000	01	34	12		LD	BC, 1234H	①
03	ED	43	00	E0	LD	(0E000H), BC	②
07	C5				PUSH	BC	③
08	01	99	99		LD	BC, 9999H	④
0B	ED	43	02	E0	LD	(0E002H), BC	⑤
0F	C1				POP	BC	⑥
10	ED	43	04	E0	LD	(0E004H), BC	⑦
14	C3	66	5C		JP	5C66H	⑧

《第3-32図》PUSH，POPの実験

① 1行目

まずBCレジスタに初期値1234Hをセットします。なお復習のため申し添えれば，これは

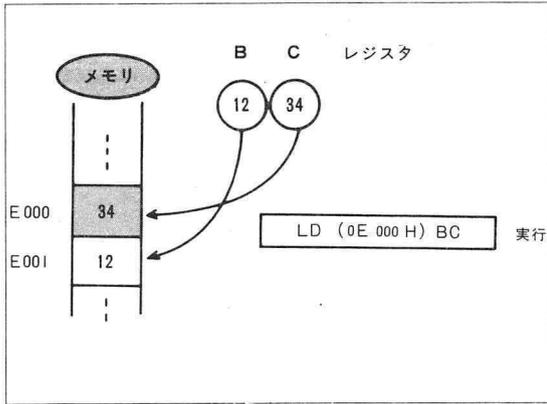
Bレジスタ＝12H
Cレジスタ＝34H

のように記憶されます。

② 2行目

1行目で本当にBCレジスタの値が1234Hになったかを確かめるため，BCの値を

E000～E001番地

に移しておきます。つまり“種もしかけもございません”というわけです。なおこの時，メモリに格納される順序に気をつけてください。**第3-33図**に示しておきます。

《第3-33図》LD, (*nn*) BC を実行すると

③ 3行目

PUSH命令実行。BCの値を保存します。

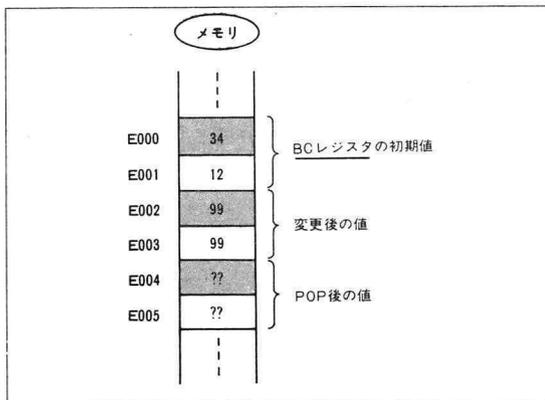
④ 4～5行目

BCレジスタの値を9999Hに変えてしまいます。それを確認するため

E002～E003番地

にBCの値を移しています。**第3-34図**を参照してください。

⑤ 6行目

POP命令実行。

《第3-34図》POP後の値は？

⑥ 7行目

果して本当にBCの値が元に戻ったでしょうか？それを確認するためBCレジスタの値を

E004～E005番地

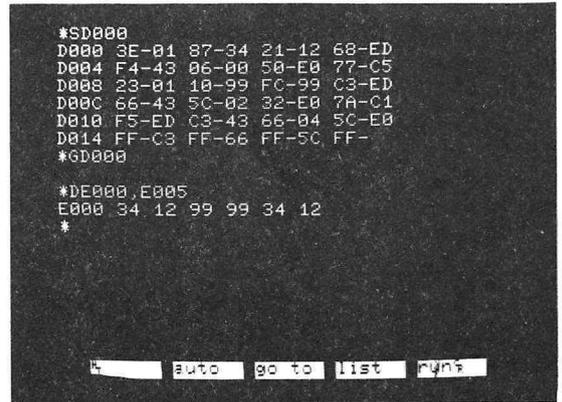
に移します。

以上が実験の主旨です。どんな結果が出るか？ さあ，**第3-32図**のプログラムを実行してみましょう。入力後，

GD000↗

でスタートです。

DE000, E005↗

でメモリの内容を確認しましょう。**写真11**をみてください。我々の期待通りの結果が出ました。

《写真11》DE000, E005でメモリ内容を確認

31. ビデオRAMポインターの調整

以上，我々は

PUSH, POP命令

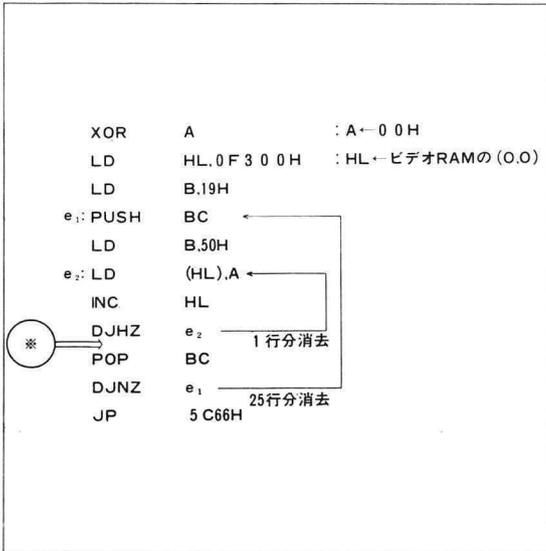
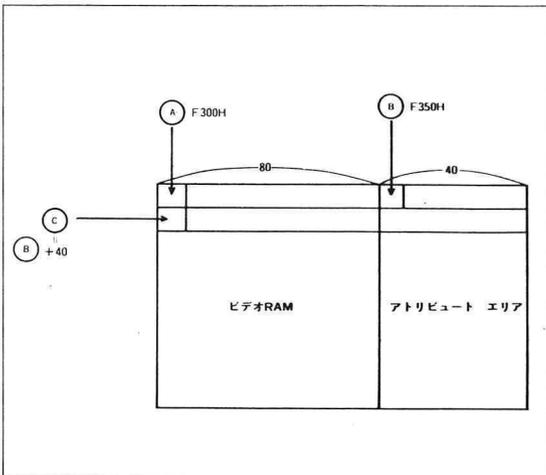
を知ったわけです。これを使うことにより，先の**2重ループの問題**を解決することができます。それは外側と内側のループに同じレジスタBを使うために起こったのでした。そこでPUSH, POPを使い

```
LD     B, XX
PUSH BC

      ∫ (この中でBの値を変えて可)

POP    BC
DJNZ  e
```

のように命令を書き，PUSH，POPにはさまれた部分にもう一つのループを入れれば解決されるのがわかるでしょう。そのことを考慮に入れて作ったのが，**第3-35図**です。これでおおむね良さそうですね。否，１ヶ所おかしいところがあるのです。

第3-36図をみてください。

《第３-35図》完成したか？

《第３-36図》ビデオRAMの構成図

最初，ＨＬレジスタはⒶ点を示しています。そして１字消去しては右隣に進みますから，これを80回繰り返し１行分の処理が進んだところで，HLレジスタはⒷ点を示すことになります。これは前に述べた**アトリビュート・エリア**（色をつけたりする部分）です。

したがって次の行の処理に入る前に，Ⓒ点まで進めてやる必要があります。

Ⓒ点＝Ⓑ点＋40

ですから，**第3-35図**のプログラムの㊟マークのところで（40＝28Hに注意して）

HL←─HL＋28H

を実行してやればよいのです。かくて我々は，

足し算の仕方

を覚える時がやってきたのです。

32. ２バイトの加算命令

２バイトの加算命令

通常２バイトの加算を行うときは，**レジスタ・ペア**を用いる。そのとき被加数はＨＬレジスタを指定すること。

〈書式〉　ADD　HL，SS

SS＝BC，DE，HL，SP

〈機能〉　HL←─HL＋SS

マシン語でアドレスの計算をするときのように**２バイトの加算を行う時は**，

ＨＬレジスタを使う

ＩＸレジスタを使う

ＩＹレジスタを使う

の三通りの方法があります。このうちもっとも良く使うのがＨＬレジスタを使う方法です。

例をあげましょう。いま

5555Ｈ＋1234Ｈ

という計算がしたいとします。それには，**二つのレジスタ・ペア**を用意します。そしてここが重要ですが，

一方は必ずHLレジスタを指定

してください。計算は，次のように行います。

```
LD   HL, 5555H
LD   BC, 1234H
ADD  HL, BC
```

HLレジスタに被加数をセットし，加数をBCレジスタ（これはどのレジスタ・ペアでも可）にセットします。そしてADD命令を実行すれば，結果がＨＬレジスタに求まります。

そこで以上の計算を実験で確かめてみましょう。**第3-37図**のプログラムをみてください。上の計算を検証できるように組み変えたものです。4行目でHLレジスタの値をみるため，その値を

E000～E001番地

にストア（格納）しています。プログラムがうまく動けば，

5555H＋1234H＝6789H

ですから，

E000H←―89H（Lレジスタ）

E001H←―67H（Hレジスタ）

のようになるはずです。

それではプログラムを入力し，走らせてみてください。その確認の仕方は――もう，おわかりですね。あとはあなたにおまかせします。

D000	21 55 55	LD HL,5555H
03	01 34 12	LD BC,1234H
06	09	ADD HL,BC
07	22 00 E0	LD (0E000H),HL
0A	C3 66 5C	JP 5C66H

《第3-37図》2バイトのADD命令

33.そして“……”が消えた

長いことかかりましたがどうやら**〈チャレンジ5〉**のプログラムも完成が近づいたようです。前節で学んだADD命令を，**第3-35図**のプログラムの㊟の位置に挿入してやればよいのです。HLに40（＝28H）を加えてやるのでしたね。

そこでザッと**第3-35図**を見渡してやります。するとDEレジスタが使われていないことがわかります。したがって最初に

DEレジスタ←―28H

をセットしておき，㊟のところで

ADD　HL，DE

とすれば良いのですね。

こうして完成したプログラムが，**第3-38図**です。さあ，せっかくですから実行してみましょう。次の順にやってみてください。

①電源ON。

②画面モードの設定

D000	AF	XOR A
01	11 28 00	LD DE,0028H
04	21 00 F3	LD HL,0F300H
07	06 19	LD B,19H
09	C5	PUSH BC
0A	06 50	LD B,50H
0C	77	LD (HL),A
0D	23	INC HL
0E	10 FC	DJNZ 0D00CH
10	19	ADD HL,DE
11	C1	POP BC
12	10 F5	DJNZ 0D009H
14	C3 66 5C	JP 5C66H

《第3-38図》画面クリア完成

WIDTH 80，25↗

CONSOLE 0，25，1，0↗

実験するには，**ファンクション・キーの内容を表示しておいた方が面白い**でしょう。

③このBACICモードの時に，デタラメで結構ですから，画面にいろいろ書き込んでおいてください。

④マシン語のコマンド・レベルに

MON↗

⑤**第3-38図**のプログラムを入力し，入力ミスをチェックする。

⑥プログラムの実行。

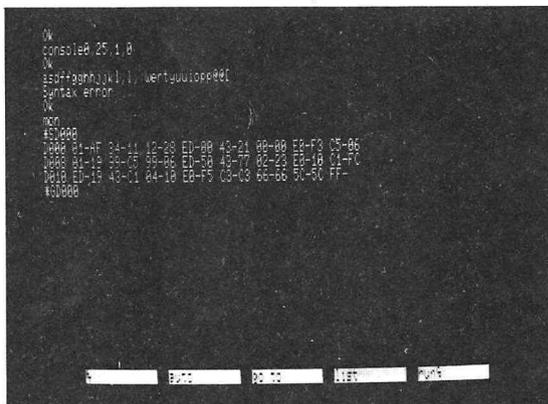
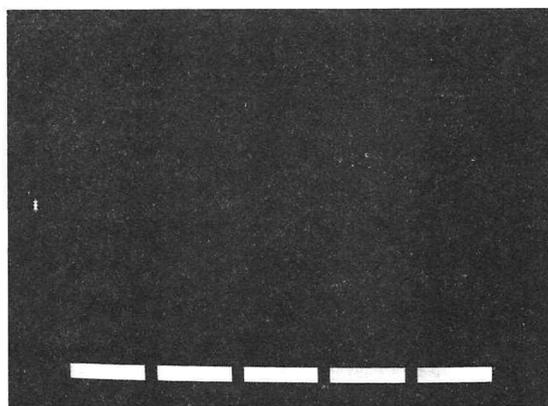
GD000↗

写真12が実行前の様子です。

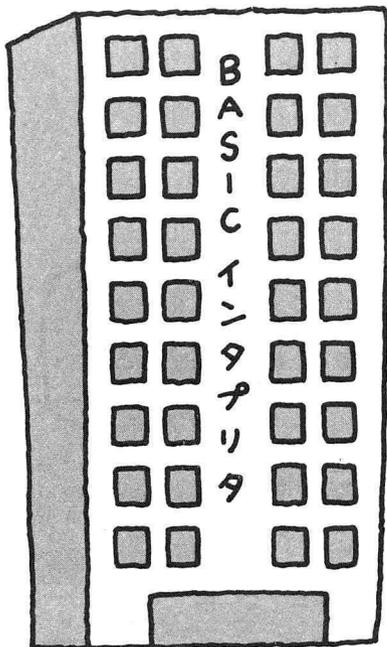
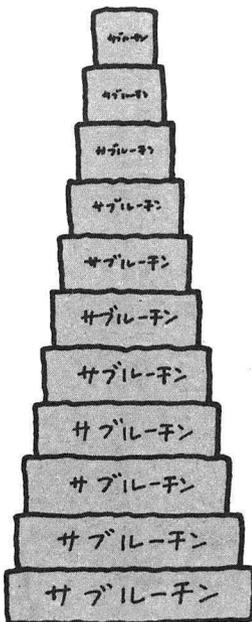
GD000

と入力してありますが，まだRETキーを押していません。そしてRETを押し，実行したのが写真13です。な，何と

ファンクションキーの表示まで消去

されてしまいました。マシン語って，恐ろしいですね。

《写真12》 プログラム実行前の様子

《写真13》 RETを押しプログラム実行

〈第３章のおわりに〉

本章は，画面消去一筋にアタックしてきたわけですが，如何だったでしょうか？　マシン語では，**たったこれだけ**のことにあれだけの手間ヒマかけているのです。するとBACICインタプリタ全体では，一体どうなっているのでしょうね。あなたの学習が進み，いつの日か力がついた時，ぜひあなたのマシンのシステムを解析してみてください。その**緻密さ**にきっと驚くことでしょう。

第4章 マシン語ユーティリティの開発

〈はじめに〉

本ブロックにおいて我々は，画面制御を中心に各種マシン語のプログラミングに挑戦してきました。本章はその集大成として

マシン語ユーティリティ(utility)

の開発に挑戦することになります。それは，これからも続くあなたのマシン語学習に大きく貢献してくれることでしょう。

本来，ユティリティは幅広い適用性が要求されますから，**どちらかというと難しい部類**に属します。しかし，いま我々はそれに挑戦するだけの実力をつけてきました。敢然とアタックしてみようではありませんか。

34. 懸案事項への挑戦

我々のマシン語に対する予備知識は，だいぶ強力になってきました。ここらで**懸案事項の解決**へとむかうことに致しましょう。

我々が初めてマシン語で組んだプログラムは，何だったかを覚えていますか？ それは，Aレジスタにある数を代入し，それを確認するためその値をメモリにストアするというものでした。第2ブロックの最初のところでやりました。そしてその後，各レジスタの値をメモリに格納してみたわけですが，それは，

実用上かなり使いずらい

ものでした。それを解決するため，今まで画面表示の勉強をしてきたわけです。おおむね画面制御もわかってきました。しからば，それを解決してみようではありませんか。

〈チャレンジ6〉

A，F，B，C，D，E，H，L

の各レジスタの値を画面に表示するプログラムを作りなさい。

ここでFについて簡単に触れておきます。

以前レジスタ・ペアの説明のとき

BC, DE, HL

の三つについては説明しましたが，実はこのF（フラグといいます）も

AとFがくっついてAF

として，あたかもレジスタ・ペアのように扱われます。“あたかも”というのは，**AFがレジスタ・ペアとして働くわけではなく，**Z－80の命令として**あたかもレジスタ・ペアのように扱われる**という意味です。具体的には，PUSH，POP命令のとき

PUSH AF

POP AF

のようにAとFがペアに用いられますが，Fは一般のレジスタのように

LD F, X

のように数値の代入はできません。

実は，F（フラグ）は**特殊な，しかも非常に重要な働き**をします（このフラグがあるため，マシン語が理解しにくくなっているのですが)。したがって，せっかくレジスタの画面表示をするなら，あとあとのことも考えて**F（フラグ）も仲間に入れてやろう**ということで，この問題にも含めました。

さあ，当面の我々の課題は**〈チャレンジ6〉**のプログラム化です。頑張りましょう。

35. プログラミングの前に

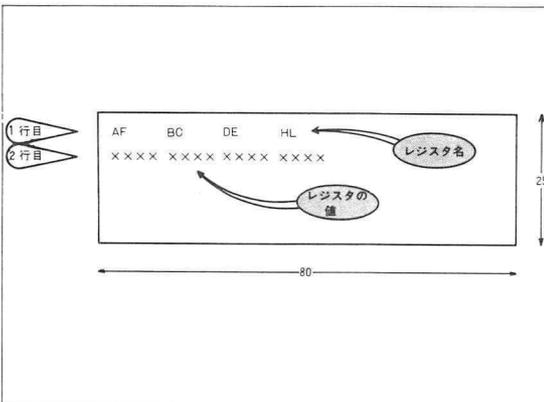
そこで作業に取りかかるわけですが，プログラミングに入る前に，画面の設計をしておく必要があります。いちおう次のように設計してみました。

画面モード：80×25

表示位置 { **レジスタ名**：画面１行目
　　　　　　　値：画面２行目

第3-39図を参考にしてください。以下，この設計にしたがってプログラミングしていくことになります。なお，これから我々のやろうとしていることは，プログラミングの分野からいえば，

システムの開発

に近い内容になります。というよりは，システム・ユーティリティの一つです。これはマシン語を学び初めて間もない人には，少しシビアな内容になるかもしれません。したがってそのことを念頭において，注意深く読んでください。

《第３-39図》画面の設計

さて，まず最初に１行目の表示を考えてみることにします。１行目の先頭から

AF　　　BC　　　DE　　　HL

と表示するわけですね。BASICなら

```
LOCATE  0, 0
PRINT  "AF    BC～"
```

でできてしまいますが，マシン語ではそう簡単にいきません。このことをマシン語で実現しようとすると，ちょっとひと工夫必要です。というのは，前章までに扱った内容と少し異なる点があるからです。

前章までの内容を思い出していただければおわかりになると思いますが，我々がこれまでに扱ったのは

同じキャラクター・コードの表示

についてです。つまり画面上にたくさんの文字を表示することはできますが，それは**すべて同じ文字**でした。ところがこれから我々がやろうとしているのは，

異なるキャラクター・コードの表示

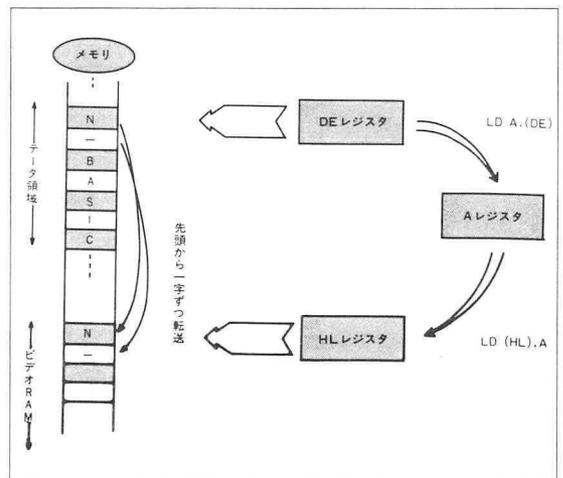
です。ところで異なるキャラクター・コードといっても1行目の場合でしたら全部で8文字しかありません。したがって単純に１字ずつ表示していってもかまわないわけです。しかし，あとあとの応用のことを考えてここでは**大量の文字列を扱う方法**を紹介しておきましょう。普通この問題を

文字列の出力

と呼んでいます。

36. "文字列出力ルーチン"のしくみ

"文字列の出力ルーチン"は必ず必要なものですから，普通どのシステムもサブ・ルーチンとして持っています。もちろんあなたのN-BASICも"文字列出力ルーチン"をシステム・サブルーチンとして内蔵しています。その仕組みは，通常次のようになっています。

第3-40図をみてください。

《第３-40図》文字列出力ルーチンのしくみ

① メモリ上の適当な位置（普通はプログラムのおわりの方）に，表示したい文字のキャラクター・コードを格納します。この部分を

データ領域（データ・エリア）

と呼びます。

② データ領域の先頭から表示したい位置のビデオR

AMに，1字ずつキャラクター・コードを転送していきます。その方法は，次の通りです。

③ DEレジスタ←―データ領域の先頭
　HLレジスタ←―ビデオRAMの転送先の先頭
のようにセットします。

④1度に
　LD (HL), (DE)
のように転送したいところですが，そういう命令はありませんから，Aレジスタ経由で2段階に転送します。

(1) データ領域のデータをAレジスタへ
　LD A, (DE)
(2) AレジスタのデータをビデオRAMへ
　LD (HL), A

⑤ 以上の④の操作で**1文字が転送されます**。次の文字を転送するには，
　INC DE
　INC HL
とやって，DE, HLの指す位置を一つ進めてからもう1度④の操作を行えばできます。

⑥ ⑤を最後のデータまで転送すれば，目的が達成されます。

以上が普通に行われている**文字列出力ルーチンのしくみ**です。これをプログラム化すれば良いのですが，**一つ気になること**がありますね。それは，

どうやって最後の文字である
と判断する？

のですか？　それをきちんとプログラム化しておかないと，あなたのマシンは最後の文字の転送が終っても，さらに文字列の転送を続けることでしょう。TV画面にはメチャメチャな文字が出現するかもしれません。

37.“文字列”を準備する

最後の文字を判定するには二つの方法があります。

① 転送する**文字の数を指定**する方法
② **ENDマーク**を用いる方法

①の方法は，すぐに理解できると思います。
　Bレジスタ←―文字数
を入れてやり，
　DJNZ命令
でループしてやれば良いのです。この方法はあなたにやっていただくとして，ここでは②の方法を紹介することに致します。

最初に文字列を用意します。**第3-41図**をみてください。“**キャラクタ・コード表**”を見ながら変換してみてください。各レジスタの間は，
　スペース＝20H
を三つ入れることにします。前の画面消去のときは，00Hを使いましたが，ここではあとで述べる**ENDマークと区別する**ため，20Hを用いています。

さて，この図の一番右側を見るとENDマークとして00Hが入っています。

ENDマーク
文字列が終了したことをプログラムの中で識別するため文字列の最後に置くマーク。ENDマークは，文字として使われないものなら，何でも可。

ENDマークについては，プログラムの作成が進むにしたがってだんだん明らかになると思います。ここでは，

文字列の最後に置かれるマーク

とでも思ってください。

以上でキャラクター・コードの準備ができあがりましたので，これをプログラムの中に割り当てることに

文字	A	F				B	C				D	E				H	L	ENDマーク
コード	41	46	20	20	20	42	43	20	20	20	44	45	20	20	20	48	4C	00

《第3-41図》文字列の準備

します。アセンブリ言語で文字列を表わすには，普通二つの方法があります。

DC……もとの文字で表わす

DB……キャラクター・コードで表わす

DCはdefine constant, DBはdefine byte の略です。これを用いて上記文字列を表わした例を第3-42図，第3-43図に示しました。マシン語にしたものはどちらも同じになりますが，アセンブリ言語の段階ではＤＣを使った方が，プログラムが見やすくなります。なお，この文字列はプログラムの最後に置く予定なので，まだ番地を割り当てていません。この段階では何番地になるか未確定だからです。

	41　46　20　20	DB　41H, 46H, 20H, 20H
	20　42　43　20	DB　20H, 42H, 43H, 20H
	20　20　44　45	DB　20H, 20H, 44H, 45H
	20　20　20　48	DB　20H, 20H, 20H, 48H
	4C	DB　4CH,
	00	DB　00H,

《第３-42図》DBを用いて

	41　46　20　20	DC　'AF　BC　DE　HL'
	20　42　43　20	
	20　20　44　45	
	20　20　20　48	
	4C	
	00	DB　00H

《第３-43図》DC, DBを用いて

38. マシン語サブルーチンの作り方

以上で文字列の準備は終りましたので，次に

文字列出力ルーチン

を作ることにします。これは，この〈チャレンジ６〉にかかわらず**汎用性のある部分**ですから，**サブルーチン**として組むことにします。サブルーチンにしておけば，他のプログラムで“文字列出力ルーチン”が必要になったとき，すぐに利用することができます。そこで**マシン語によるサブルーチンの作り方**を覚えることになります。

> **CALL命令**
> **〈書式〉**：CALL　番地
> **〈機能〉**：サブルーチンを呼ぶ
> **RET命令**
> **〈書式〉**：RET
> **〈機能〉**：サブルーチンからメインルーチンに戻る。

以上の命令を使ってサブルーチンを作るわけです。これらの命令とBASICの命令を対応させてみましょう。

〈BASIC〉		〈マシン語〉
GOSUB	←→	CALL
RETURN	←→	RET

これでおおむね使い方は，おわかりになったと思います。次のように使っていただければ結構です。

```
            ∫
    CALL   X番地        ┐
            ∫          ├ メインルーチン
    JP     5C66H       ┘
X番地：                 ┐
            ∫          ├ サブルーチン
    RET                ┘
```

39. フローチャートで

そこで以上のような書式で**“文字列出力”のためのサブルーチン**を作ることにします。このサブルーチンは，メインルーチンで

ＬＤ　ＤＥ，文字列の先頭の番地―――①

ＬＤ　ＨＬ，ビデオRAMの番地――――②

のように各番地をセットしてから

CALL　××××H

のように呼ぶことにします。したがって①，②の値を変えることによって**いろいろな文字列**を**ＴＶ画面の任意の位置**に表示することができます。

以上の入力条件のもとにこのサブルーチンの中身を考えると，**第3-44図**のようになるでしょう。

①　ＤＥレジスタの指す文字列のところから，データ

をAレジスタに移します。

《第3-44図》"文字列出力ルーチン"フロチャート

② ここで**文字列の終りの判定**をします。

もし全ての文字列をビデオRAMに転送し終っていれば，**DEレジスタはENDマークの00を指しています**から，①の命令によってENDマークの00がAレジスタに転送されているはずです。そこで

Aレジスタ＝00？

を判定します。そして

Aレジスタ { ＝0：サブルーチンおわり
　　　　　 ≠0：まだ転送中

と考えれば良いのです。

③ まだ文字列の転送がおわっていなければ，Aレジスタに格納された出力すべきコードを，HLレジスタの指すビデオRAMに転送します。

④ DEの値を一つ大きくします。これによりDEレジスタは，文字列の次の文字を指すことになります。

⑤ HLの値を一つ大きくします。これによりHLレジスタは，ビデオRAMの次の位置を指すことになります。

⑥ ①に戻します。

以上で文字列の最終の判断の仕方——ENDマークの使い方についておわかりいただけたことと思います。そこでここまでをプログラミング化してみたいと思います。ただし，以上の処理の中で②の

Aレジスタ＝00？

の判定については，まだ我々の知識ではプログラムできません。これについては，

マシン語を学ぶうえでの最難関
‖
フラグ

をマスターする必要があります。

40. 二つの重大なフラグ

それでは，最難関の**フラグ**に挑戦していくことに致しましょう。あなたは，すでに

フラグ

という言葉に何回かお目にかかったことがあると思います。ここで再び**"レジスター覧表"**を参照してください。"アキュームレータとフラグ"の欄に

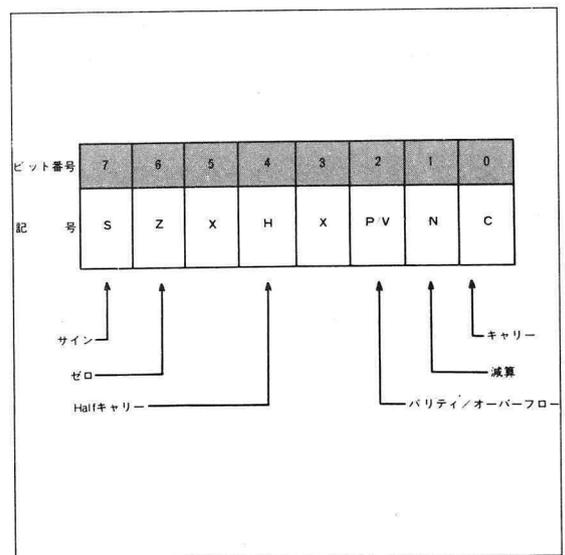
F（フラグ）

というのがありますね。それによると，

フラグ＝8ビット・レジスタ

であることがわかります。8ビット・レジスタですから，

○○○○　○○○○

のように8つの記憶場所（ビット）を持っているはずですね。フラグの場合，この個々のビットに個有の名前がついています。それは，次の通りです（**第3-45図**参照)。

《第3-45図》フラグ　レジスタ

ビット番号　名　前

0：キャリー・フラグ
1：減算フラグ
2：パリティ／オーバーフロー・フラグ
3：未定義
4：Halfキャリー・フラグ
5：未定義
6：ゼロフラグ
7：サイン・フラグ

まず，ここまで納得してください。

OKでしたら，次に進みます。以上のように

フラグ・レジスタは8ビット・レジスタであり，八つのビットのうち六つについては特定の名前がついていて，それぞれ××フラグ

と呼んでいます。これら六つのフラグのうち，私は，

二つだけ

マスターすれば十分と考えています。それは，

ゼロ・フラグ
キャリー・フラグ

の二つです。たとえば，やがて我々が解析しようとしている“**スペース・インベーダー**”は，この二つのフラグしか使っていません。したがって，当面はこの二つのフラグのマスターに全力を上げてください。他の四つについては，将来余力ができた時で十分です。

41. フラグはいつ変化する

さて，各フラグはビットですから

0　または　1

の値を取ります。それでは，各フラグの値は**いつ変わる**のでしょうか？　また，**どのようにして0か1か**決まるのでしょうか？

答。いつ―――しょっちゅう。
0と1は―――実行した命令と演算結果により決まる。

これだけでは，良くわかりませんね？

マシン語のプログラムをRUNさせると，CPUはマシン語の命令を一つ一つ実行していきます。そして一つの命令を実行するたびに（例外はありますが）各フラグの値は変化します。

どのフラグが変化するか？
0になるか1になるか？

は，命令によって決っています。それは付録の

“命令のフラグへの影響”

をご覧になってください。そしておおむね次のように考えて結構です。

① **ゼロ・フラグの場合**

ゼロ・フラグは普通，**Zフラグ**とか**Z**と略したりします。ある演算を行ったとき

演算結果＝0⟶Z＝1
演算結果≒0⟶Z＝0

になります。ところで一般にフラグの値が1になることを，その**フラグが立つ**と表現しています。この言葉を使えば，

ある演算を行ったとき
演算結果＝0
になるとフラグが立つ！

と言えます。

② **キャリー・フラグの場合**

キャリー・フラグは略して**CYフラグ**，**CY**，あるいは単に**C**と書いたりします。キャリー・フラグの場合は，次の二つに分けて考えるのが良いでしょう。

(1) **加算系の命令の場合**

おおむね足し算に類する命令を行った結果，

桁あふれをおこした―――**CY＝1**
桁あふれをおこさなかった―――**CY＝0**

のように変化します。ここで**桁あふれ**とは，次のようなことを指します。たとえばいま計算に8ビット・レジスタを用いたとします。ところで

8ビット＝0～255

までの数を表わせますね。さて

8ビット＋8ビット

の計算を行った結果，必ずしも答が

0～255

に収まるとは限りません。計算によっては255を越える――すなわち8ビットでは表わせないこともあるのです。これを**桁あふれが生じた**と呼んでいます。同様に16ビットのレジスタを用いた場合，結果が16ビットで表わせないとき，すなわち

16ビット＝0～65535

を越えたとき，**桁あふれが生じた**と呼ぶわけです。

(2) **減算系の命令の場合**

この場合は簡単です。ある加算系の命令を実行

した結果,

答がマイナスになった——CY=1

答がプラスまたは0———CY=0

となります。

以上が，Ｚフラグ，ＣＹフラグの大体のあらましです。次に以上のことをもう少し具体的に説明してみたいと思います。

42. フラグを観察する

それでは**第3-46図**により，フラグの変化する様子を観察してみることに致しましょう。この中には，我々のまだ知らない命令も含まれていますので，合わせて説明することにします。

命　　令	意　　味	Aレジスタ	Zフラグ	CYフラグ
LD　A,32H	A←50	50	—	—
ADD　A,64H	A←A＋100	150	0	0
ADD　A,6EH	A←A＋110	4	0	1
SUB　05H	A←A−5	255	0	1
ADD　A,01H	A←A＋1	0	1	1

《第3-46図》フラグの変化する様子

① まず表の見方は，次のとおりです。

一番左側が**実行した命令**です。次がその**命令の意味**を説明したものです。ここに登場する五つの命令は，すべて8ビットの演算を対象にしており，演算結果はＡレジスタに格納されます。したがって3番目が**命令実行後のＡレジスタの値**を示しますが，これイコール**演算結果**となります。したがってこの欄を見れば，フラグがどう変化するかわかります。

② まず**1行目**。ＬＤ命令でＡレジスタに32Ｈをセットしています。

32Ｈ＝50

に注意してください。ＬＤ命令では，フラグの値は変化しません。したがって

Ｚフラグ，ＣＹフラグ

共にそれ以前の値を保持しています。これを表では“—”で表現しています。

③ **2行目**。これは加算命令です。Ａレジスタに

64Ｈ＝100

を足しています。Ａレジスタの値は50でしたから，

Ａレジスタ←—50＋100＝150

になります。そこでフラグの値を考えましょう。演算結果は150で0ではありませんから，Ｚフラグは立ちません。すなわち

Ｚフラグ＝0

に変化します。また加算命令で演算結果が255を越えていませんから，

ＣＹフラグ＝0

となります。

④ **3行目**　さらに

6ＥＨ＝110

を加えます。演算結果は，

150＋110＝260

で255を越えますから，キャリー・フラグが立ち，

ＣＹフラグ＝1

となります。結果が0ではありませんから，やはり

Ｚフラグ＝0

です。さて260は8ビットに収まりません。こんな時，**Ａレジスタの値はどうなるのでしょう？**

8ビットの演算で桁あふれをしたとき

〈公　式〉

Ａレジスタ←—答　−256

この公式より，

260−256＝4

ですからＡレジスタの値は4（04Ｈ）になります。

⑤ **4行目。SUB命令**——これは，Ａレジスタの値から数を引き，結果をまたＡレジスタにしまう命令です。Ａレジスタは4でしたから

4−5＝−1

で演算結果がマイナスになりましたから，

ＣＹフラグ＝1

と立ちます。0ではありませんから

Ｚフラグ＝0

です。さて答がマイナスになったとき，Ａレジスタにはどのように格納されるのでしょうか？

8ビットの演算でマイナスになったとき

〈公　式〉

Ａレジスタ←—答　＋256

この公式により，

−1＋256＝255

がAレジスタに格納されます。

⑥　5行目。ADD命令は，Aレジスタに1を加える命令です。

255＋1＝256

で255を越えましたから，

CY・フラグ＝1

です。またAレジスタには

256－256＝0

により，0が格納されます。したがって

演算結果＝0

と見なされ，

Zフラグ＝1

となります。

以上，CPUが命令を実行するたびに各フラグがパカパカ変化するのがおわかりいただけたと思います。

43. 黙って坐ればフラグで判定

フラグの意味，漠然とながらおわかりいただけたでしょうか？　まだ使い方まではタッチしていませんから，曖昧模糊としているかもしれません。しかし，ここらで本線に戻りたいと思います。我々は，**“文字列出力ルーチン”**を作っているところでした。そしてENDマークの判定，すなわち

Aレジスタ＝00H？

の判定をしようとしていたのです。

この判定をするにはいろいろな方法があります。ここでは

Zフラグ

を使って判定してみようと思います。それには，わざとダミーの演算を行ってみます。演算を行えば，

フラグが変化

します。その結果

Zフラグが立った――――ENDマーク

Zフラグが立たない――まだ終りでない

のように判定しようというわけです。

それには，どのような演算を行えば良いでしょうか？次の条件を満たす必要があります。

〈演算の条件〉

Aレジスタの値を変えてはいけない

これはわかりますね。つまり，

①　**まだ文字の出力が終っていないとき**

Aレジスタには，これから表示しようとするキャラクタ・コードが入っています。これはもちろん，00ではありません。したがってAレジスタの値を変えない命令を実行しても，

Aレジスタ≠00

ですから，Zフラグは立ちません（Z＝0）。

②　**ENDマークに達したとき**

Aレジスタの値は00になっています。ここにAレジスタの値を変えない命令を実行しても

Aレジスタ＝00

と変化しませんから，Zフラグが立ちます(Z＝1)。

話がだいぶ具体化してきました。もう一度**第3-44図**のフローチャートをみてください。◇の条件判断のところでAレジスタの値を変えない命令を実行します。その結果

Z=1――RET命令実行

Z=0――ビデオRAMへの転送を続ける

のようにプログラミングすれば良いのです。あとは

Aレジスタの値を変えない命令

を捜せば良いのですね。

44. 変えない，変えないAレジスタ

Aレジスタの値を変えない命令って何だと思いますか？　実は，これがたくさんあるのです。もっとも理解しやすいのは，次の二つでしょう。

ADD　A，00H

SUB　00H

0を足しても引いても値は変わりませんから，これは明らかですね。しかし，ここではもっともポピュラーな別の方法を御紹介致します。

それは前ブロックで登場した“古き時代のハイ・テクニック”のところで取り上げた

論理演算

を使う方法です。したがってこれから紹介する方法も，どちらかというと**有名な“古き時代のハイ・テクニック”**になるでしょう。それは，次の命令です。

AND A

AND命令

〈書式〉：AND S
S＝A, B, C, D, E, H, L,
(HL), (IX＋*d*),
(IY＋*d*), *n*

〈機能〉：AレジスタとSとの論理積を計算し，結果をAレジスタに格納する。

それでは，

AND A

でAレジスタの値が変化しないことを，具体的に調べてみましょう。**第3-47図**に論理積の真理値表を掲げておきますので，それを見ながら確認してください。なお

AND A

の場合，

AレジスタとAレジスタの論理積

をとるという意味です。

		AND
0	0	0
0	1	0
1	0	0
1	1	1

上表のように
0 AND 0 ＝0
1 AND 1 ＝1
であるから、同じ値の論理積をとっても、値が変化しない

《第3-47図》論理積真理値表

① **Aレジスタ＝00Hのとき**

2進数に変換してから論理積をとるのでしたね。

A：0000　0000
A：0000　0000 **(AND)**
　　0000　0000

ご覧のように

0　AND　0＝0

ですから，すべてのビットが0で，これを16進数に戻しても00Hで変わりません。

② **Aレジスタ≠00Hのとき**

ここでは，“♥”のキャラクター・コードで確認してみましょう。

♥のキャラクター・コード＝E9H

また，

E9H＝1110　1001

ですから，

A：11110　1001
A：11110　1001 (AND)
　　11110　1001

となり，やはりAレジスタの値は変わりません。

45. “文字列出力ルーチン”のプログラミング

以上のように

AND A

を実行しても，Aレジスタの値は変わりません。しかし，**論理演算といえども立派な演算ですから，フラグの値は変化**してくれます。したがって

AND A

を実行後，フラグの値を調べることでAレジスタの値が00H，すなわちENDマークであったかを判定できるわけです。

以上，フラグの意味，使い方を大体マスターしていただけたことと思いますので，ここで**“文字列出力ルーチン”**を**第3-44図**のフローチャートにしたがって作ってみようと思います。

第3-48図をみてください。これができあがったプログラムです。大体おわかりになると思いますが，2ヶ

1	E04A	1A	LD　A, (DE)
2	4B	A7	AND　A
3	4C	C8	RET　Z
4	4D	77	LD　(HL), A
5	4E	13	INC　DE
6	4F	23	INC　HL
7	50	18　F8	JR　0E04AH

《第3-48図》文字列出力ルーチン

所説明を加えておいた方が良さそうなところがあります。

① **1行目**

文字列からAレジスタにコードを取ってきます。

② **2行目**

Aレジスタどうしの論理和をとり，Zフラグを変化します。

③ **3行目**

ここは，**RET**命令の使い方が重要です。RET命令は，先に見ましたようにBASICのRETURNと同じようにサブルーチンから戻るのに使います。しかし，BASICと異なり，次の二つの使い方があります。

〈書式1〉　RET

〈書式2〉　RET（条件）

ここで（条件）とは，フラグの値のことで，Zフラグ，CYフラグについては次の4種類があります。

C───CY＝1のとき
NC───CY＝0のとき
Z───Z＝1のとき
NZ───Z＝0のとき

この条件が満たされたときだけRET命令が実行されます。またもし条件が満たされないときは，RET命令は実行されず，その下の命令に制御が移されます。

そこで3行目の命令を見てみると

RET　Z

になっています。これは条件付きのRETで，

Z＝1

ならばRETするという命令です。したがって**ENDマークを発見するまで**は，下の4行目の命令に進むことになります。

④ **4行目**

ENDマークでないと，Aレジスタのコードをビデオ RAMへ転送します。

⑤ **5行目**

文字列のポインタDEを一つ進めます。

⑥ **6行目**

ビデオRAMのポインタHLを一つ進めます。

⑦ **7行目**

これは初めての命令ですね。

JR……相対ジャンプ

と呼ばれる命令で，我々の知識の範囲で書けば，

JP　0E04AH

と同じ意味になります。これについては，次節でまとめることにします。

以上のようにこのサブ・ルーチンは，全体が一つのループになっていて，**一つのループを繰り返すたびに1文字ずつ表示**していきます。これで大体“文字列出力ルーチン”は，理解いただけたのではないでしょうか？

46. 相対ジャンプの計算

前節の補足として，**相対ジャンプ**の説明を加えておきたいと思います。

JP　0E04AH───①
JR　0E04AH───②

上の①，②はまったく同じ命令です。

①──**絶対ジャンプ**
②──**相対ジャンプ**

と呼び方を変えて両者を区別しています。②の相対ジャンプ命令は，Z－80の前身である8080にはありませんでした。ということは，②が①の**改良版である**ということです。一言で言えば，

> 相対ジャンプを使うには，**条件**がある。そして**その条件を満たす限りは，相対ジャンプを使う方がbetter**である。なぜなら絶対ジャンプに比べ，相対ジャンプの方が**二つ長所**があり，**短所は一つ**しかないから

となります。このことは，すぐ説明しますが，①，②のどちらを使うかは好みの問題でいますぐ両方を覚える必要はありません。

まず相対ジャンプが使えるための条件から。これは

〈条　件〉

相対ジャンプはその命令の前127バイト，うしろ128バイトしかジャンプできない

ということです。次に，

〈長　所〉

①リロケータブルなプログラムが作れる。
②省メモリ。

このうち①については，我々の現在の知識ではまだ理解不能です。②については，感覚的にすぐわかると思います。①の命令をマシン語に変換すると**3バイト**になりますが，②の命令では**2バイト**にしかなりません。

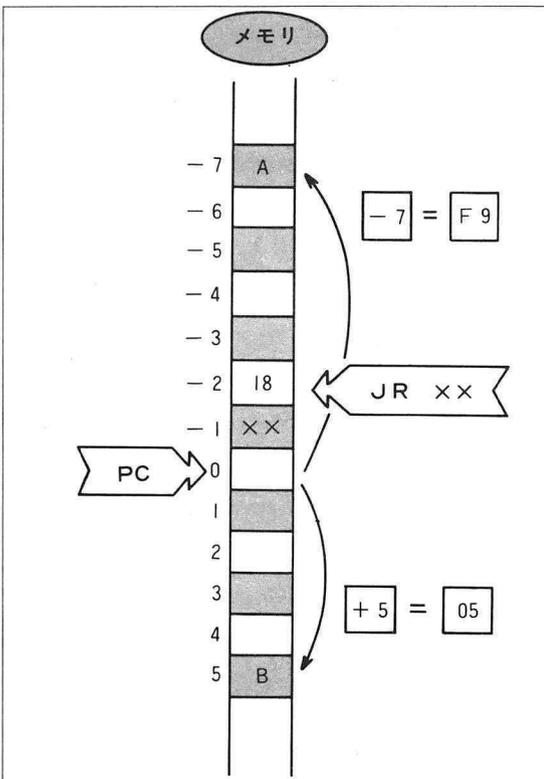
以上，**相対ジャンプの特色**を見てきましたが，おそらく聡明なあなたは，

DJN Z

を思い出したのではないでしょうか。相対ジャンプをマシン語に変換するには，やはりDJNZと同様の方法を用います。まずＪＲは，次の２バイトに変換されます。

１８———ＪＲの命令で個有

××———ＰＣからの距離で不定

最初の18Hは，ＪＲであることを表わし，いつも同じです。次の１バイトは，どこにジャンプするかを表わし，これはジャンプ先によって変わります。その変換の仕方は，**第3-49図**をご覧になってください。

《第３-49図》"相対ジャンプ"をマシン語に変換

真中あたりにＪＲ命令を表わす18Hがあります。ＰＣは，DJNZのところで触れた**プログラム・カウンタ**です。次の命令の位置を示すのでしたね。したがってＣPUがJR命令を読み込んだ直後は，PCは図の位置を指すことになります。ジャンプ先を計算するには，この

PCの位置を基準＝0

にするのでしたね。たとえば，図のＡ地点にジャンプさせるには，－７。またＢ地点にジャンプさせるのでしたら，＋５ですね。もちろんこのままではマシン語になりませんから，付録の**"符号付き16進数表"**を見ながら16進数に変換します。それぞれ

Ｆ９，05

になります。

以上の説明をもとに，**第3-48図**の７行目を自分でハンド・アセンブルしてみてください。

18　F8

になりましたか？

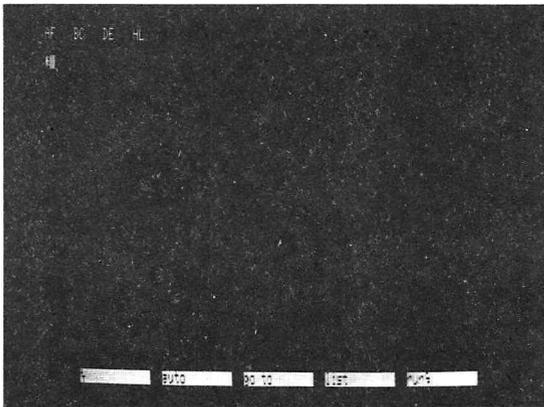
47. なつかしい文字列に再会

前節までで"文字列出力ルーチン"が完成しましたのでうまく動くか実験してみましょう。

最初は，**〈チャレンジ６〉**のために**第3-43図**で用意した文字列を表示してみたいと思います。次の順で実験してみてください。

① 電源ON
② 画面モードを，80×25にセットする。
③ 第3-50-①図のメイン・ルーチンをキー・インする。
④ 第3-48図のサブルーチンをキー・インする（メイン・ルーチンとサブルーチンとの間に隙間があきますが，別に支障はありません)。
⑤ **第3-42図**の文字列を，上のサブルーチンに引きつづいてＥ052番地からキー・イン。
⑥ GE000↗

でプログラム・スタート！

写真14のように予定通りの画面が得られたことと思います。いちおう簡単にメイン・プログラムを見ておきましょう。

《写真14》第3-39図の画面が得られた

1行目は，DEレジスタに文字列の先頭をセットしています。2行目は，HLレジスタに表示すべき位置をセットしています。この二つのパラメータをセットしたあと3行目で，“文字列表示ルーチン”を呼び，表示が終ると4行目でモニタにジャンプし，プログラムを停止しています。

もう一つ実験してみましょう。

第3-50図②のプログラムを見てください。これは，**第3-50図①**のメインプログラムのうち，**文字列パラメータDE**の値を変えています。したがって最初の実験と異なる文字列が表示されることになります。さあ，何が表示されるのでしょうね。さっそくプログラムを直して走らせてみましょう。

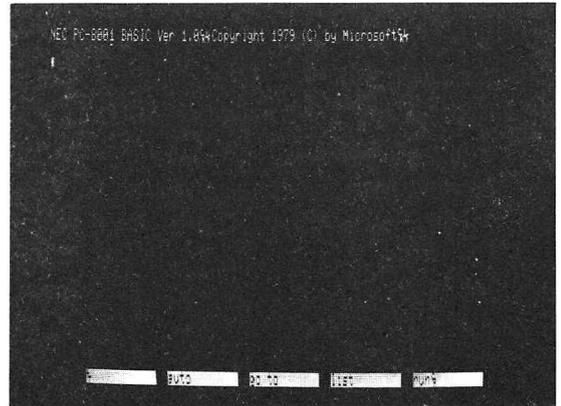
E000	11	52	E0	LD	DE, 0E052H
03	21	00	F3	LD	HL, 0F300H
06	CD	4A	E0	CALL	0E04AH
09	C3	66	5C	JP	5C66H

《第3-50図》① 文字列出力ルーチンの実験①

E000	11	38	18	LD	DE, 1838H (注)
03	21	00	F3	LD	HL, 0F300H
06	CD	4A	E0	CALL	0E04AH
09	C3	66	5C	JP	5C66H

(注)この部分のみ異なる。

《第3-50図》②“文字列出力ルーチン”の実験②

写真15のようになりましたね。1行目に表示されたのがそれです。どこかで見たことのあるメッセージですね。そうです。これは，**N-BACICスタート時のメッセージ**です。実は，ROM内の1838Hにはこのメッセージの文字列が格納されているのです。しかし，**写真15**を見ると少し変ですね。

《写真15》ROM内の1838Hの文字列

普通このメッセージは，2行に分けて表示されます。しかし，写真を見ると1行に表示されていますね。しかも途中に

$^{C}{}_{R}$ とか $^{L}{}_{F}$

の記号が見られます。何だ，コリャ？

簡単に言えば，これらもPCのキャラクター・コードの一つです。“キャラクター・コード表”をご覧になってください。

$^{C}{}_{R}$ ……… 0DH
$^{L}{}_{F}$ ……… 0AH

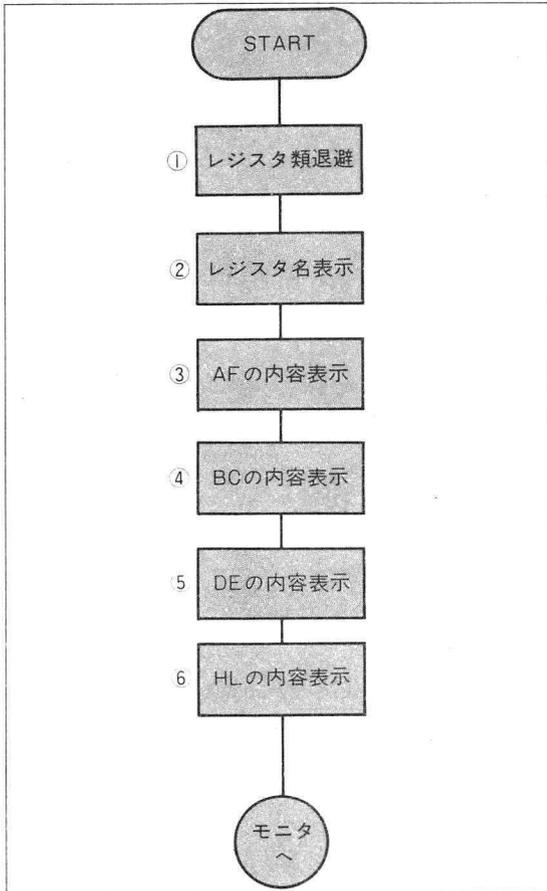
となっています。これらは，**特殊記号**と呼ばれ，キーボードから入力することはできません。PCのROM内にある“文字列表示ルーチン”は，いま我々が作ったものとは少し異っています。通常のキャラクター・コードの時は同じように画面に表示しますが，**特殊文字の時は，異なった動作**をします。たとえば，

$^{C}{}_{R}$ $^{L}{}_{F}$

がくると改行するように作ってあります。したがって現在の文字列を，**ROM内の文字列表示ルーチン**を使えばこの文字列が正しく表示されるのです。ちなみにROM内のそのルーチンは，52ED番地から始まっています。**第3-50図**の2行目を「21 38 18」，3行目を「C3 ED52」に変えて実験してみてください。なつかしい文字列が，正しく表示されることでしょう。

48. 鳥瞰図

“文字列出力ルーチン”は，大切な汎用サブルーチンですから，少し脱線して説明してみました。先へ進みましょう。次は，“レジスタ表示”の本体にかかわる部分です。ヨイショ。

ここから少し複雑になりますので，最初に全体の方針を見てみましょう。**第3-51図**のフローチャートをみてください。

《第3-51図》“レジスタ表示プログラム”フロチャート

① この**“レジスタ退避”**というのは，いわゆる

PUSH命令

のことです。各レジスタの値を表示する前に，いろいろな処理が行われますが，その際にレジスタの値が変化してしまいますから，プログラムの1番最初のところでPUSH命令を実行し，レジスタの値をスタック領域に保存しておきます。

② この部分は，**“文字列表示ルーチン”**によりレジスタ名を表示するもので，既に前節まででコーディングが終っています。

③ AFの内容を表示します。

④ BCの内容を表示します。

⑤ DEの内容を表示します。

⑥ HLの内容を表示します。

すべての表示が終ったら，モニタにジャンプさせ，プログラムを終了します。

以上がプログラム全体の鳥瞰図です。以下，このフローチャートにしたがって未完成の部分を見ていくことにします。

49. アセンブラによるリストを読む

第3-52図をみてください。これが,〈チャレンジ6〉の解答である**“レジスタ表示プログラム”の完成版**およびそれを**テストするためのプログラム**です。ところでこのリスト，我々が今まで見てきたものとは少し異なりますね？

我々は，本書の最初のところで**ハンド・アセンブルの方法**を学びました。そのときアセンブル作業をするのに何も手作業に頼らなくとも，

アセンブリ言語──→マシン語

の変換を自動的に行ってくれるプログラムがありますよと述べておきました。それがいわゆる

アセンブラ

と呼ばれるものであることは言及しておきました。ちなみにPC—8800では，簡単なアセンブラがROMに内蔵されています。**第3-52図**のプログラム・リストは，実は**アセンブラ**によって出力されたものです。あなたは本書をここまで読み進めてこられ，マシン語にかなり慣れ親しんだものと思われます。そして今後更に学習を進めていくうえで，**アセンブラによるリスト**に接する機会が多くなると思います。そのときのためにここでアセンブラのリストに慣れていただければと思い，**第3-52図**に見本を出品した次第です。と言うのは，アセンブラによるリストには**アセンブラ特有の記号**が使われていますから，それを知らないと読みにくいからです。そこで以下に**アセンブラのリストの見方**をまとめておきたいと思います。

① **ORGとEND**

これらは**“アセンブラ指示語”**または**“擬似命令”**と呼ばれているものの一つです。これは，他のアセンブリ言語と異なり，マシン語には変換されません。プログラマーがアセンブラそのものに対して指示する命令です。**第3-52図**を見ると**ORG**は，プログラムの最初に，また**END**は一番最後にありますね。これはアセンブラに対し，

ORG：プログラムのスタート番地
END：プログラムの最後

を指示しているのです。

② **EQU命令**について。

ORGの次には，EQUというのが見られます。

この命令を理解するには，**ラベル**という概念を知る必要があります。

> **ラベル**（label）
> アセンブラ（assembler）において特定の番地につけた名前。定義をしておけば，アセンブリ言語でアドレスの代わりに記述できる。

たとえば，**第3-52図**の2行目を見ると，

MON：EQU　5C66H

と書いてあります。これは，

5C66番地にMONという名前をつける

という意味です。このように定義しておけば，あとでコーディングするとき，

JP　　5C66H

と書くかわりに

JP　　MON

と書くことができ，非常に便利です。

③ ラベルを定義する方法は，もう一つあります。リストのE000番のところを見てください。

MAIN：PUSH　HL

とあります。ここは，単に

PUSH　HL

と書くこともちろん可能です。ところがこのように書くと，**自動的に**

E000番地＝MAIN

のようにラベルの定義がなされます。わざわざ

MAIN：EQU　0E000H

と書く必要がなくなるわけです。

④ 以上のようにアセンブラでは，ラベルというものを使うことができ，ハンド・アセンブルに比べ**プログラムの開発が容易**になっています。そのラベルの定義の仕方は以上見てきたように二つの方法があります。その使い分けは，次のようにしてください。

定義する番地が
プログラム内にある——③の方法
プログラム内にない——②の方法

以上，アセンブラによるリストを読む際の注意を列挙してみました。これだけわかれば，あとは実物（たとえば **第3-52図**）とにらめっこしていくことにより除々にリストの見方がわかってくると思います。

《第3-52図》レジスタ表示プログラム　アセンブル・リスト

```
************************
* << ﾚｼﾞｽﾀｰ ﾋｮｳｼﾞ >>   *
*           1982.3.11  *
*          BY K. ﾂｶｺﾞｼ  *
************************
                ;
                        ORG   0DFF5H
                ;
5C66            MON:    EQU   5C66H
F300            LINE1:  EQU   0F300H
F378            LINE2:  EQU   0F378H
                ;
DFF5 3EAA       TEST:   LD    A,0AAH        ;SET REG for TEST
DFF7 01CCBB             LD    BC,0BBCCH
DFFA 11EEDD             LD    DE,0DDEEH
DFFD 213412             LD    HL,1234H
                ;
E000 E5         MAIN:   PUSH HL             ;ﾒｲﾝ ﾙｰﾁﾝ
E001 D5                 PUSH DE
E002 C5                 PUSH BC
E003 F5                 PUSH AF
E004 1152E0             LD    DE,HED        ;HEDER
E007 2100F3             LD    HL,LINE1
E00A CD4AE0             CALL PR
E00D 2178F3             LD    HL,LINE2      ;ﾚｼﾞｽﾀｰ ﾋｮｳｼﾞ
E010 C1                 POP  BC
E011 CD23E0             CALL WRD
E014 C1                 POP  BC
E015 CD23E0             CALL WRD
E018 C1                 POP  BC
E019 CD23E0             CALL WRD
E01C C1                 POP  BC
E01D CD23E0             CALL WRD
E020 C3665C             JP   MON
                ;
E023 78         WRD:    LD    A,B           ;BC=REG,HL=VRAM
E024 CD2DE0             CALL BYTE
E027 79                 LD    A,C
E028 CD2DE0             CALL BYTE
E02B 23                 INC   HL
E02C C9                 RET
                ;
E02D F5         BYTE:   PUSH AF             ;A=REG,HL=VRAM
E02E C5                 PUSH BC
E02F 0604               LD    B,4
E031 0F         BY1:    RRCA
E032 10FD               DJNZ BY1
E034 C1                 POP  BC
E035 CD39E0             CALL HALF
E038 F1                 POP  AF
                ;
E039 E60F       HALF:   AND   0FH           ;A(ｶｲ4ﾋﾞｯﾄ),HL=VRAM
E03B FE0A               CP    10
E03D 3804               JR    C,HA1
E03F C637               ADD   A,55          ;A-F
E041 1804               JR    HA2
E043 CBEF       HA1:    SET   5,A           ;0-9
E045 CBE7               SET   4,A
E047 77         HA2:    LD    (HL),A
E048 23                 INC   HL
E049 C9                 RET
                ;
E04A 1A         PR:     LD    A,(DE)        ;DE=DATA,HL=VRAM
E04B A7                 AND   A
E04C C8                 RET   Z
E04D 77                 LD    (HL),A
E04E 13                 INC   DE
E04F 23                 INC   HL
E050 18F8               JR    PR
                ;
E052 41462020 HED:      DC    'AF   BC   DE   HL'
E056 20424320
E05A 20204445
E05E 20202048
E062 4C
E063 00                 DB    0
                ;
E064                    END
```

←ORG：アセンブラ指示語。スタート番地の指示。

←EQU：ラベルの定義。

←アセンブラ指示語。プログラムの終りを示す。

50.メイン・ルーチンの解析

それでは，第3-52図のアセンブル・リストを解析して行きましょう。

DFF5～DFFF番地。

この部分は，あとで実験に使うためのもので直接“**レジスタ表示プログラム**”とは関係ありません。

E000～E003番地。

レジスタ類の退避です。**第3-51図**フローチャートの①の部分ですね。特に問題はありませんね？

E004～E00C番地

“文字列表示ルーチン”をCALLして，TV画面の一行目にレジスタ名を表示しようとしているところです。

E00D～E01D番地

問題のレジスタを表示する部分です。この部分を良く見ると，

CALL　WRD

という命令が四回出ているのがわかります。WRDというのは，ラベルです。つまり

WRD＝E023番地

のサブルーチンを四回CALLして処理しているのです。四回というのは，

AF, BC, DE, HL

のようにレジスタ・ペアの数ですから，WRDを呼ぶたびに，**一つずつレジスタ・ペアの値が表示される**のだろうと予想されます。

E020番地。

モニタへジャンプします。

以上がメイン・ルーチンの解析です。あとは，

サブルーチンWRD

の使い方がわかれば，すべてが解決します。しかし，それには，

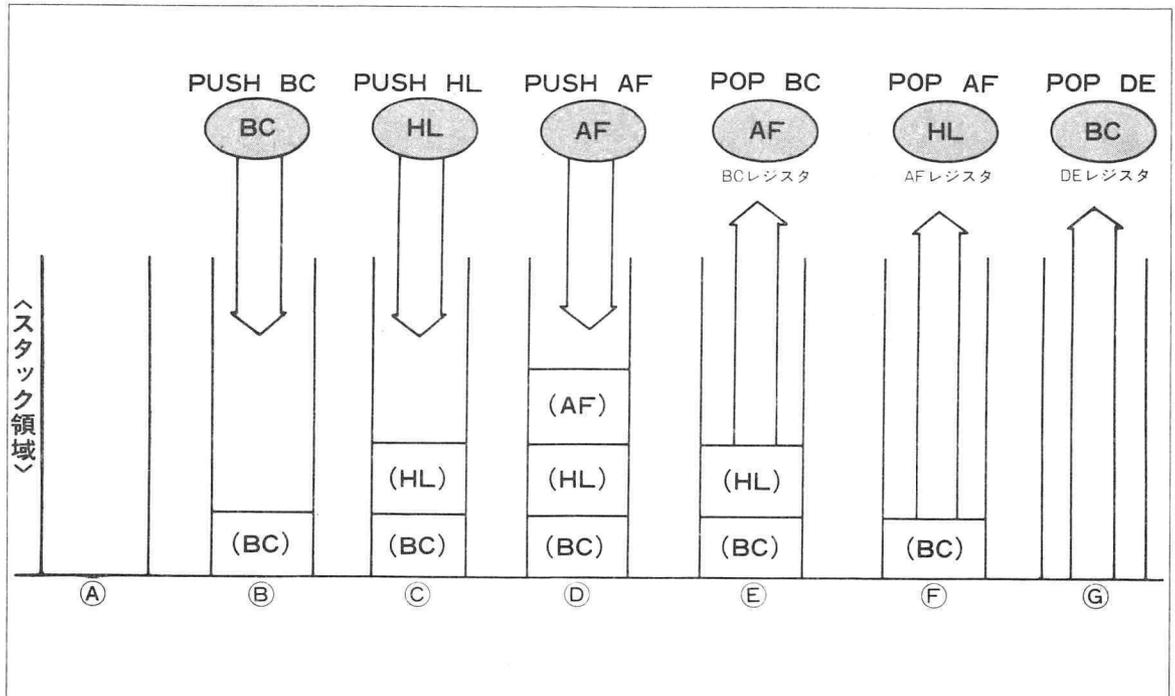
スタック命令の入出力順序

についての予備知識が必要です。

51.ラスト・イン・ファスト・アウト

E000～E003番地のように複数のレジスタ・ペアの値をスタック領域に保存した時には，その値をスタック領域から取り出す際，**順序に注意**する必要があります。

スタックを取り扱う時は，

ファースト・イン・ラスト・アウト

《第3-53図》ラスト・イン・ファスト・アウト

または

ラスト・イン・ファースト・アウト

の原則にしたがいます。要するに，

1番最後に入れた値が先に出てくる

ということです。

このことを**第3-53図**で見てみましょう。Ⓐのような円筒形を考え，これが**スタック領域**だと思ってください（スタック領域は，本当はメモリ上にあるのでしたね)。最初はカラです。

Ⓑで

PUSH　BC

を実行すると，ＢＣレジスタの値がスタックの底に保存されます。続いてⒸ，Ⓓのように

PUSH　HL

PUSH　AF

と実行していくと，各レジスタの値がスタックに積まれていきます。

次にⒺでスタックの値を取り出すために

POP　BC

を実行したとします。するとＢＣレジスタに，一番最後にスタック領域に入ったＡＦの値が得られます。続いてⒻで

POP　AF

を実行すれば，ＡＦレジスタにスタック領域の一番上にある前のＨＬの値が入ります。最後にⒼで

POP　DE

を実行すれば，ＤＥレジスタに残りの旧ＢＣの値が入るわけです。

以上のように**POP命令**を実行すると，その**レジスタ名に関係なく，**

スタック領域の最新の値

を得ることができるわけです。

ラスト・イン・ファスト・アウトの原則

(last in first out)

納得していただけましたか？

52. WRDの機能

そこでサブルーチンＷＲＤを見て行きます。

サブルーチンWRD

〈入力条件〉：ＢＣ＝表示する値

HL＝ビデオRAMの番地

〈出力条件〉：HL＝次の表示位置

〈機　能〉

BCレジスタの内容をHLで示される位置から表示する。

サブルーチンWRDは，以上のような機能を持っています。**第3-52図**のプログラムのＥ00Ｄ番地を見てください。HLレジスタにTV画面の２行目のアドレスをセットしています。次の

POP　BC

では，ＢＣレジスタに**何の値**が入るでしょうか？　先の“ラスト・イン・ファスト・アウト”の原則を思い出してみてください。Ｅ003番地で１番最後にスタック領域に入ったAFの値が，BCにセットされるのです。そして

CALL　WRD

でまずAFの値が画面に進みます。

次にBCの値を表示するわけですが，今度は**HLレジスタの値はセットする必要はありません。**なぜならWRDを実行すると，自動的にHLの値が次の表示位置まで進んでくれるのでしたね。ですから単に

POP　BC

でスタック領域の次の値を取り出し，

CALL　WRD

とすれば表示することができます。

以下，同様に

DE, HL

の各レジスタ・ペアの値を表示することができます。

53. WRDの中身

次に，WRDの中身を調べてみます。

Ｅ023～Ｅ02Ｃ番地を見てください。WRDはさらに下位のサブルーチンBYTEをCALLすることで成立しています。

サブルーチンBYTE

〈入力条件〉：Ａ＝表示する値

HL＝ビデオRAMの番地

〈出力条件〉：HL＝次の表示位置

〈機　能〉

Aレジスタの内容をHLレジスタで示される位置から表示する。

このサブルーチンBYTEを利用するわけです。WRDの目的は，BCの2バイトの内容を表示することにあります。そこでそれをBとCの二つに分け，まずE023番地でAにBの値を移して

CALL　BYTE

でAの内容を表示します。次にE027番地でAにCの値を移し，表示すれば良いわけです。BとCの表示が終ったところでHLレジスタは，いま表示した4桁の16進数の右隣を指しています。そこで次の表示とくっつかないようにするためE02B番地で

INC　HL

を実行して1字分あけてやります。

以上が，サブルーチンWRDの中身です。

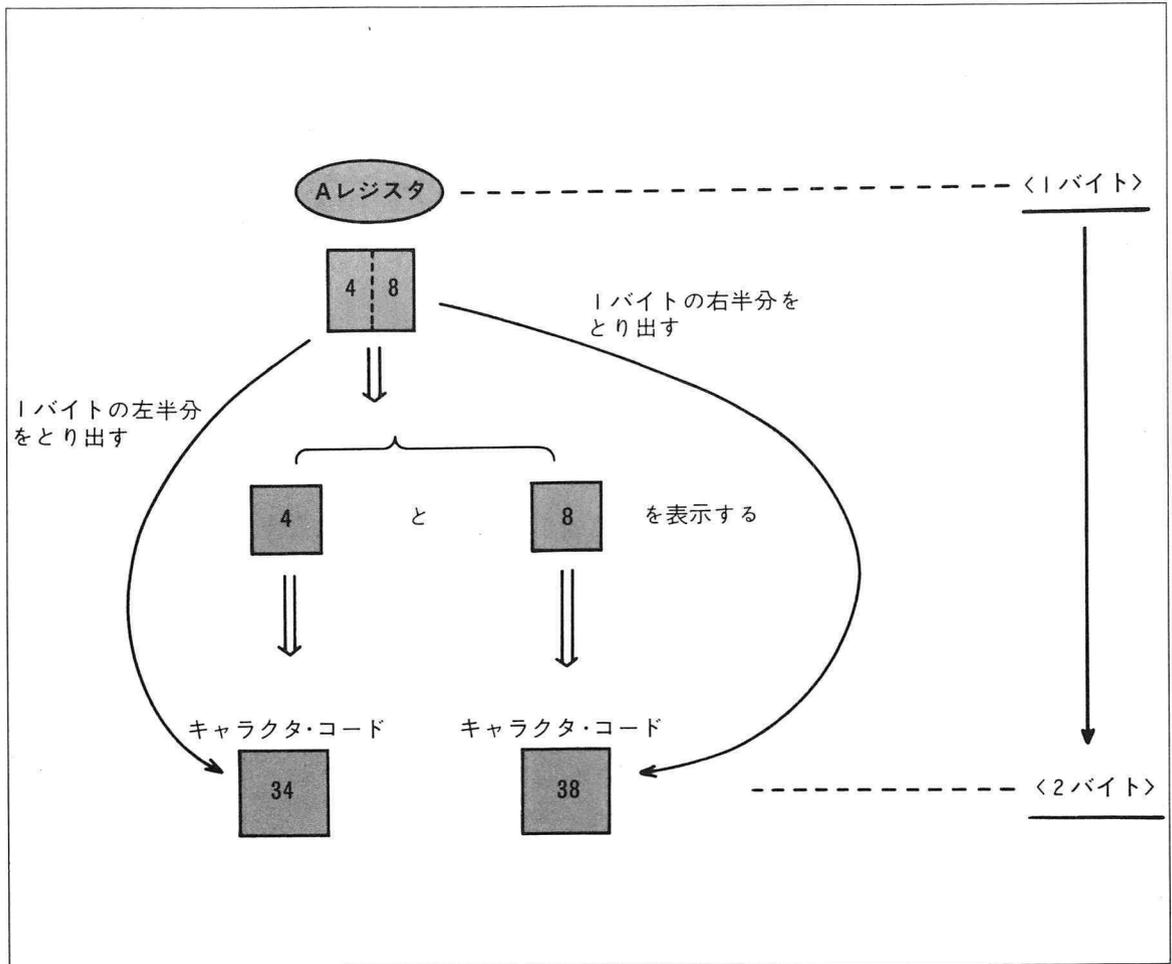
54. 1バイト――基本的構成単位

次に下位のサブルーチンBYTEを解析します。ここは少し難しいかと思いますが，頑張って読破してみてください。

サブルーチンBYTEの目的は，**Aレジスタの内容をTV画面に表示**しようとするものです。これは，1見簡単なことのように見えますが，次に示しますようにひと工夫必要です。

第3-54図をみてください。仮に

Aレジスタ＝48H

だったとしましょう。すると“4”と“8”の2字を表示すれば良いわけですから，それぞれをキャラクター・コードに変換し，

《第3-54図》 1バイトを分解する

4 ──→34H

8 ──→38H

の2バイトをビデオRAMに転送すれば良いのです。これを原理的に見れば,

①Aレジスタの1バイトを

4　と　8

に分解する。

②それぞれの頭に3をつける。

34　　38

これで二つのキャラクター・コードができ上がります。ところで，我々のマシンは**8ビット＝1バイト**のマシンですね。ですから**1バイトが基本**で

1バイトを二つに分解することは不可能

なわけです。つまり基本的には，上のような原理はできないことになります。困りましたね。

55. BYTEを解析する

こんな時，**ビット単位**に働く命令を用いて解決することができます。以下，順に説明して行きます。

E02D～E02E番地。

Aに入っているこれから表示しようとする値，及びBCに入っているレジスタ・ペアの値が破壊されないようにスタックに入れて保存しておきます。

E02F～E033番地

ここでビット操作の命令を施します。まずRRCA命令を理解してください。

> **RRCA命令(Rotate Right Circular A)**
> Aレジスタの値を右に1ビット回転する。
> CYフラグにビット0の値が入る。

第3-55図で説明致しましょう。最初Aレジスタの値が,

0011　0010

であったとします。ここに

RRCA

を実行すると，1ビットずつ右にズレます。ビット0はそれ以上右に行けませんから，ビット7にまわります。また，ついでにCYフラグにもその値が入ります。

さて、E02F～E033番地を見るとDJNZ命令によってこの**RRCAが4回繰り返されている**のがわかります。RRCAを4回も繰り返すとどういうことが起こるでしょうか？　上位の4ビットがそっくり下位4ビットに回わってしまいます（もちろん下位4ビットは上位に回わりますが)。たとえば最初

A＝32H

だったとしたら,

A＝23H

になるわけです。

E034～E037番地

BCレジスタの値をもとに戻してから，サブルーチンHALFをCALLします。

サブルーチンHALF

〈入力条件〉：A＝表示したい数
　　　　　　　HL＝ビデオRAMの番地

〈出力条件〉：HL＝次の表示位置

〈機　能〉

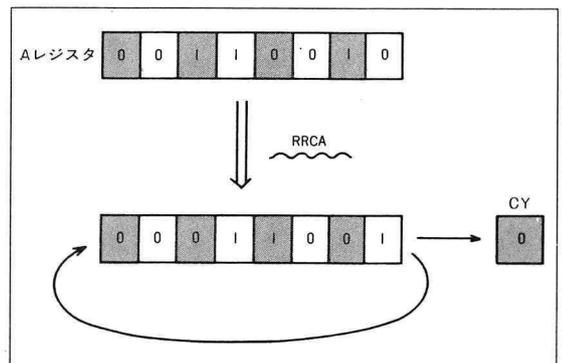
Aレジスタの下位4ビットの数をHLで示される位置に表示する。

HALFは，以上の機能を持ちますからこれで最初の数が表示されることになります。

E038番地

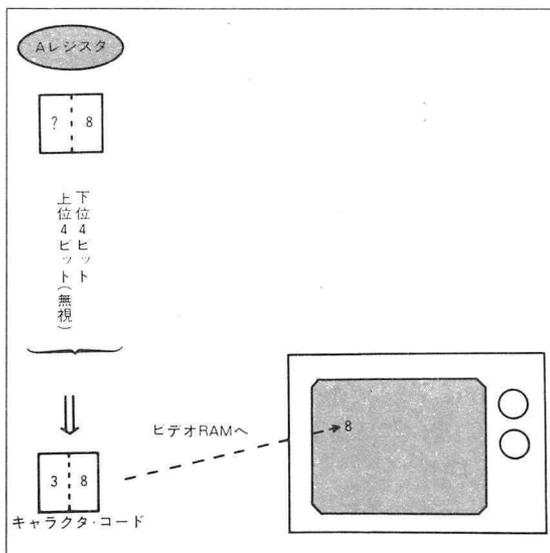
POP　AF

を実行していますから，Aレジスタの値が回転前のものに回復しています。そしてそのままサブルーチンHALFに抜けていますから，これで2字目がめでたく表示されることになります。

以上，サブルーチンBYTEを解析してきました。少し難しかったですか？　**あと一つ，HALFを解析すれば,〈チャレンジ6〉が完成**します。頑張れ，頑張れ！フー，フー。

《第3-55図》RRCA命令

56. ビット操作を使って

最後のHALFです。

このルーチンの仕事は，第3-56図のようにAレジスタの下位４ビット（16進数は，４ビットで１桁になるのを覚えていますか？）を，TV画面に表示することです。

《第３-56図》HALFの機能

そこで **"キャラクター・コード表"** をご覧になってください。変換するのに，次の二つのケースに分れるのがわかります。

① **0—9の数字のとき**

上位４ビットを３にしてやれば良い。

② **A—Fの英字のとき**

これは少しわかりにくいので，10進数で考えてみましょう。まず変換前に上位４ビットを０に変えてしまいます。すると変換前は，

0A～0F

のいずれかになりますから，10進数で表わせば

10～15

です。次に変換後のキャラクター・コードを10進数で表わせば，

A＝41H＝65

B＝42H＝66

C＝43H＝67

⋮

となります。変換前と変換後を比べてください。

〈変換前〉＋55＝〈変換後〉

になります。これで方針が決まりました。

以上の予備知識をもとに，HALFの部分を解析していくことに致しましょう。

① **E039番地**

論理演算を行っていますね。これは，

不要な上位４ビット＝０

にしているのです。なぜなら

0FH＝0000　1111

です。この上位４ビットは，すべて０ですから相手が何であろうとANDをとれば０になります（真理値表を思い出してください）。また下位４ビットはすべて１です。

（1　AND　0）＝0

（1　AND　1）＝1

により，下位４ビットは相手が何であろうとも値が変わりません。

② **E03B～E03E番地**

これは，Aに入っていた値が

0～9：数字

A～F：英字

の判定をするものです。これを理解するには，新しい命令CPを知る必要があります。

> 比較命令CP
>
> 〈書式〉：CP　n
>
> 〈機能〉：Aレジスタの値とnを比較し，その結果をフラグに入れる。
>
> $A<n$ →CY＝1
>
> $A=n$ →　Z＝1

この**CP命令**を使い，**フラグの値**により

小さい，等しい，大きい

の判定ができます。E03B番地では，

Aレジスタの値　と　10

を比較していますから，

Aが数字（00～09）⟶CY＝1

Aが英字（0A～0F）⟶CY＝0

となり，CYフラグで判定できます。

ところで以前，**RET命令にフラグの条件を加味できることを述べましたが，**JPやJRの**ジャンプ命令にもフラグの条件を加味できます。**したがってE03

D番地は，

CY＝1

すなわち数字のときはHA1にジャンプしなさいということです。

ここでは，**もう1点，注意**を申しあげておきます。E03B番地は，本来は

CP　0AH

と書くべきでしょう。しかしアセンブラの場合，リストのように10進数で書いても，きちんと16進数に変換してくれます。

③　E03F～E042番地

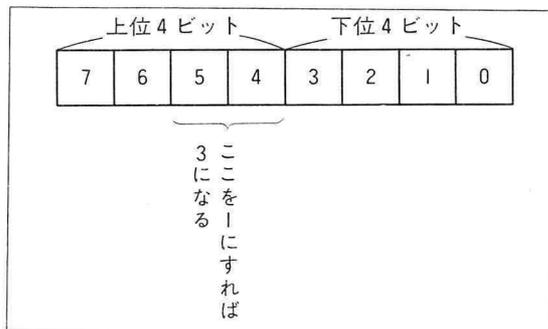
英字の場合の処理です。55を足してキャラクター・コードに変換し，HA2へジャンプします。

④　E043～E046番地

数字の場合の処理です。現在上位4ビットは，0になっています。それを3に変えてやるのでしたね。ここに出てくるSET命令も，ビット操作に関するものです。2進数で上位4ビットを考えてみましょう。

0H＝0000

3H＝0011

したがって4ビットと5ビットを1にしてやれば，上位4ビットを3に変えることができます（ビットの数え方は，1番右を0ビットとします。（**第3-57図**）。そこでSET命令です。

《第3-57図》ビットの数え方

SET

〈書式〉 SET　b，X

b＝ビット番号

X＝レジスタ，メモリ

〈機能〉：指定したレジスタ，またはメモリの指定したビットを1にする。

なおここではSET命令を使いましたが，他の命令や論理演算を使ってもできます。考えてみてください。

⑤　E047～E049番地

Aは既にキャラクター・コードになっていますから，ビデオRAMに転送してやります。HLを一つ右に進めて処理を終了します。

いかがですか？　これで“レジスタ表示プログラム”のすべての解析が終りました。あとは，これを走らせるだけです。

57. 試　運　転

このプログラムは，ただ走らせただけでは正しく動いたかわかりません。そこで**第3-52図**のプログラムでは，**前の方にテスト用のデータを用意**しました。

DFF5—DFFF

の部分がそれです。

A——AAH
B——BBH
C——CCH
D——DDH
E——EEH
H——12H
L——34H

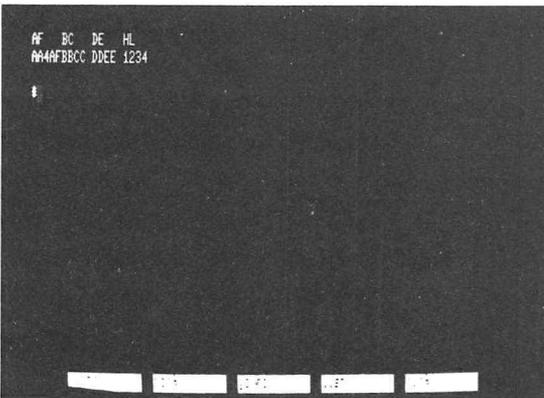
のようにセットしてあります。そしてそのままE000番地からの“レジスタ表示プログラム”に飛び込みますから，これらの値が表示されるはずです。

それでは，プログラムをキー・インして走らせてみましょう。なおこのプログラムは，

1行80字モード

にしておかないと正しく動きませんので注意してください。

GDFF5↗

で走らせます。**写真15**のようになれば成功です。

《写真15》試運転の結果

〈チャレンジ6〉のおわりとして，**“レジスタ表示プログラム”**についてまとめておきましょう。

このプログラムは，単なる使い捨てのプログラムではありません。それはあなたがこれからマシン語をやって行く上で**強力な武器**となるでしょう。しかもあなたは，その**すべてを解析**しました。したがって今後あなたの使いやすいように改良していくこともできるのです。

まず，

WE000,E063↗

で**今回のプログラムをSAVE**しておきましょう。そしてあなたがマシン語をいじくっていく上で，レジスタの中身が見る必要が起きたとき，

①このプログラムをLOAD。
②あなたのプログラムで，レジスタを表示したいところを

JP　　0E000H

に変える。
③あなたのプログラムを走らせる。

以上の手続きで，その時のレジスタの値が表示されるはずです。

〈第4章のおわりに〉

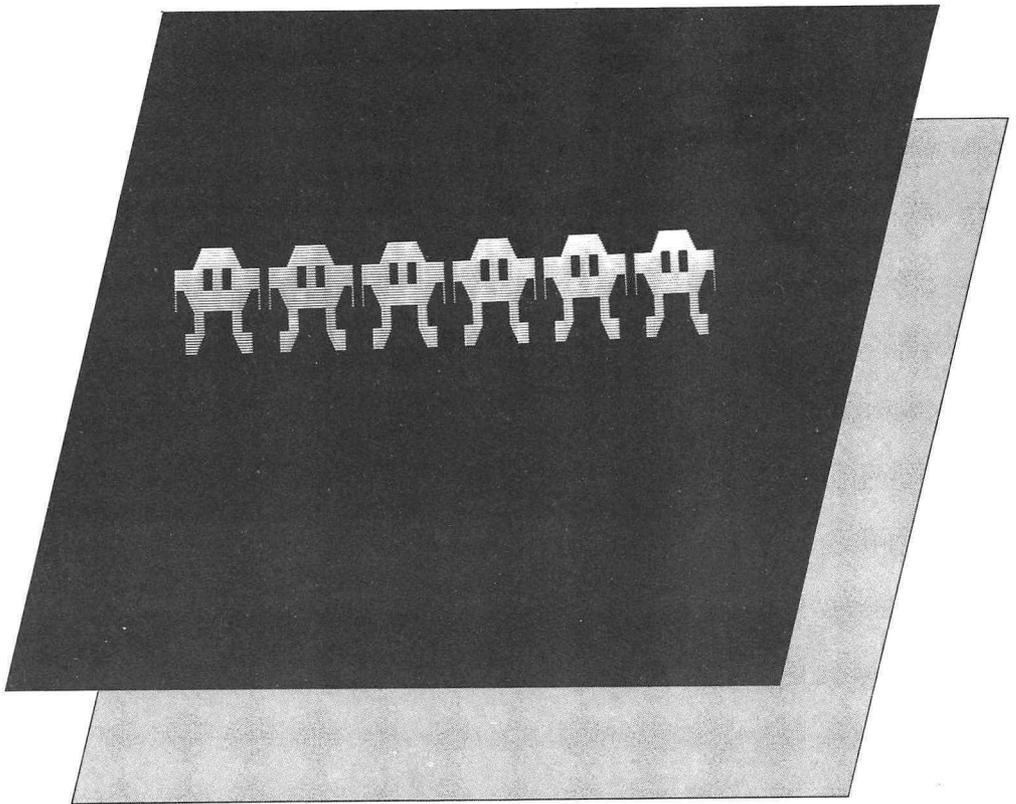
マシン語ユーティリティの開発，いかがだったでしょうか？　以上をもちまして本書の中心を成す長かった第3ブロックも終了となります。不幸にして理解しにくい部分がありましたら，何回も読み返し，ぜひ

完全に吸収

してください。そして，いつの日かあなたの目差したプログラムに挑戦できるよう学習を続けてください。ご成功をお祈りしています。

というわけで，次は本書最後のブロックとなります。そこには，オール・マシン語版“スペース・インベーダー”でさえ製作できるよう，インベーダー（侵略者）が登場してきます。さてどういうことになりますか——。

第4ブロック

マシン語の散歩道

き・ら・く・に　歩いてみよう………

前ブロックまでの長い長い**マシン語基礎への旅**，ご苦労様でした。すでにあなたは，たくさんのマシン語の知識を身につけています。そんなあなたに本書の最後としてこのブロックを用意しました。

本ブロックは，学習のためのブロックではありません。気楽に“**マシン語の散歩**”を楽しんでください。

本ブロックには，二つの章が用意してあります。最初の章は，さらに詳しい**画面制御への指針**が示してあります。それらは，本書の続編で詳しく展開されるでしょう。それまであなたに気楽に**カラー・グラフィック**を楽しんでいただくための案内です。

第2章は，**キャラクター**の表示です。やがて本書のIII編で**オール・マシン語版カラー・グラフィック**による“**新しい精密なゲーム**”を解析していくことになりますが，この章がそれの超マイクロ版といえるでしょう。

いずれにしても気楽に，気楽に読み進めていってください。

第1章 さらに詳しく知りたい人のために

1. Z-80命令の種類

今まで何度かハンド・アセンブルしてきましたが，1行をマシン語に変換すると，

1〜4バイト

の四つに分れるのに気がついていたでしょうか？　本節ではこのあたりのことをまとめておきましょう。

第4-1図をみてください。

Z-80の命令を分類すると，正確には

5種類

に分けられます。

マシン語の1バイト目，または2バイト目までを

オペレーション・コード

(operation code)

または略して**OPコード**と言います。これは，命令の意味を表わす部分です。

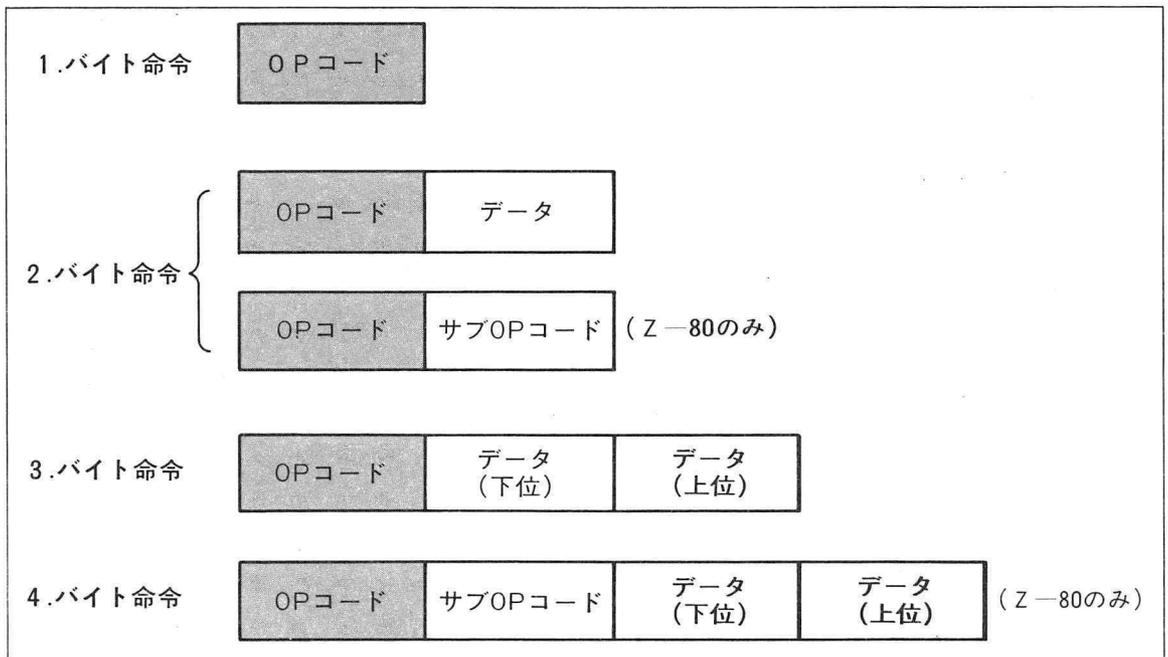
命令の中には**OPコードだけで成立**するもの（たとえばRET）と，**OPコードを修飾する部分が必要なもの**（たとえばJP　5C66H）の2種類があります。この修飾する部分を

オペランド(operand)

と言っています。以上を言語学の言葉を借りていえば，次のようになるでしょう。

自立語：OPコードのみで成立

他立語：OPコード＋オペレーションで成立

なお表中，“Z-80のみ”というのは，Z-80CPUで追加された命令で，その前身である8080CPUにはなかった命令です。

《第4—1図》Z—80命令の種類

2.( )のアル・ナシ

次は,( )のアル・ナシについてまとめてみたいと思います。
あなたは，次の命令の違いについてハッキリと説明ができるでしょうか？

LD　HL,１２３４H
LD　HL,(１２３４H)

見た目には，**ソックリ同じ**です。違いは

( )があるかないか

だけです。気分的な違いだけでしょうか?

むろん**両者は，まるっきり違います**。たとえば，ハンド・アセンブルしてみると，

LD　HL,１２３４H
　　⟶**21**　３４　１２
LD　HL,(１２３４H)
　　⟶**2A**　３４　１２

のように**OPコードが異なる**ことでもわかります。これについては前節で簡単に説明したことはあったのですが，次の実験を通してその違いをハッキリと認識しておきましょう。

〈実　験〉
①第３ブロック，**第3-52図**のプログラムをロードする（カセットに収めてありますね?）。
②**第4-2図**のプログラムを入力する。
③ＧＤ０００↗
　を実行する。
④**第4-3図**のプログラムを入力する。
⑤ＧＤ０００↗
　を実行する。

D000	21	34	12	LD	HL.1234H
03	C3	00	E0	JP	0E000H

《第4-2図》LD　HL,1234Hを実験する

D000	2A	34	12	LD	HL,(1234H)
03	C3	00	E0	JP	0E000H

《第4-3図》LD　HL,(1234H)を実験する

《第4-4図》実験結果

それでは，順に説明して行きます。
まず③を実行した場合，**第4-4図**①のように表示されるでしょう。図中××の部分は，そのときのレジスの値により不定です。今は，**HLレジスタの値**を問題にしていますから，あまり気にする必要はないでしょう。
また，この実験は必ず

１行80字モード

で実験をしてください。なぜなら**第3-52図**の"レジスタ表示プログラム"は，１行80字モードで制作したからです。
さて，今の実験結果から，

LD　HL,１２３４H

では，

HLレジスタ⟵１２３４H

のように代入されることがわかりました。
次に⑤の実験結果です。今度は,**第4-4図**②のようになりましたね。すなわち，

LD　HL,(１２３４H)

では，

HLレジスタ⟵２８０１H

のように代入されています。結果が全然異なりますね？
それでは，いったい

２８０１H

とは何でしょう?

それには，次の手続きをしてみればわかります。

D1234, 1235↗

つまり，1234番地およびその次の1235番地の値を見てみるのです。すると，

1234番地：01――Lに代入された

1235番地：28――Hに代入された

であることがわかります。つまり

LD　　HL，(1234H)

を実行すると，

1234Hではなく

1234Hの中身

が代入されるのです。

(　)は，その中身を示す

のに使われていたのです。

どうですか？　これで今までモヤモヤしていたことがハッキリしたのではないでしょうか？　以上を住所にたとえてみれば，

(　)がなければその**住所を**

(　)があればその**住民を**

表わしており，

両者はまったく異なる

と言えます。

3. さらに詳しい画面制御

我々は，前ブロックにおいてマシン語による画面制御を見てきたわけですが，まだまだ物足りないと考える人がいるかもしれません。たとえば**カラー・グラフィック**をやりたい人は，まだ前ブロックだけの知識では困難かもしれません。これらをきちんと説明しようとすると，まだまだかなりのページ数を必要とします。したがってそれらについては本書の続編に譲るとして，ここではそれまで自分でいろいろ実験してみたい人のために，その指針を簡単に御紹介しておきましょう。

① **画面モードを1行40字で使いたい人のために**

これは，簡単な実験でその方法を知ることができます。まず**BASICのコマンド・レベル**で

WIDTH　40↗

を実行して画面モードを整えてください。しかる後に**MON**でマシン語のコマンド・レベルにし，

SF300↗

でビデオRAMにデータを書き込んでいきます。こうしてしばらく実験していれば，ビデオRAMのどの番地が使われているかわかってくるでしょう。

1行36字モードとか，他のモードでも実験してみてください。

② **グラフィックを使ってみたい人のために**

本格的にやろうとすると，やはり**アトリビュート・エリアに関する知識が必要**となってきます。ここではもっと簡単にグラフィックを楽しむ方法を紹介しておきます。

まずBASICのコマンド・レベルで

LINE (0, 0) ― (159, 99)

, PRESET, BF↗

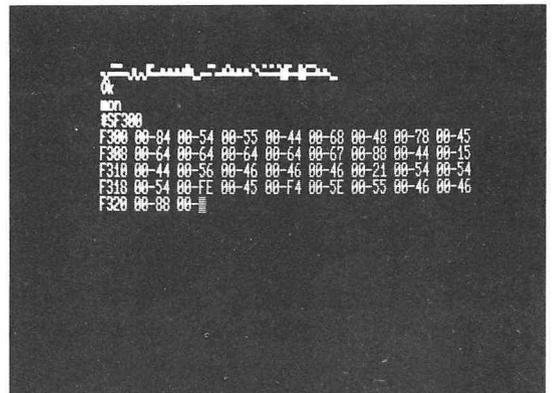
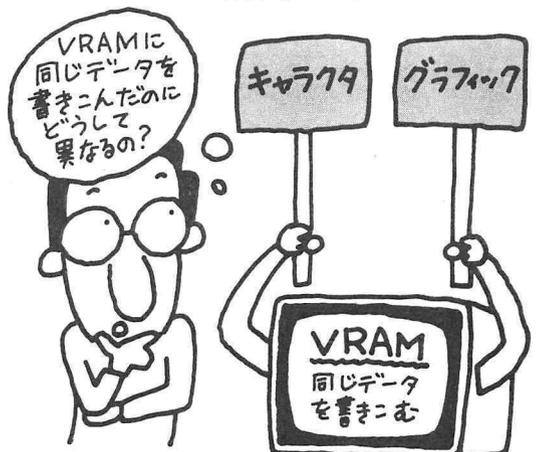
を実行します。そして“OK”のメッセージが出るまでしばらく待ちましょう。“OK”が出たら

MON↗

SF300↗

です。あとは，キャラクターを表示する要領でデタラメなデータを書き込んでいってください。**写真1**のように

TV画面にグラフィックが出現

するでしょう。

《写真1》TV画面にグラフィック出現

さてキャラクター・コードに対するグラフィック・コードの作り方も心要ですね。それには，**第4-5図**の“**グラフィック・パターン変換図**”を使います。グラフィック・パターンの具体的な作り方を，**第4-6図**で説明致しましょう。

(1)　図のようなグラフィック・パターンを作りたいとします。

(2)　黒い部分を1，白い部分に0を当てはめます。

(3)　“**グラフィック・パターン変換図**”を見ながらデータを**ビット番号順**に並び換えます。

(4)　これを8ビットの2進のデータとみて，16進数2桁に変換します。

A5H

これができあがったグラフィック・コードです。

以上の方法でできあがったグラフィック・コードをビデオRAMに転送すれば，自由にグラフィックをあやつることができます。なおキャラクター・モードに戻したかったら，制御をBASICコマンド・レベルに戻し，

LINE（0，0）—（79，24）
　　，“　”，BF↗

を実行してください。

③　**カラー・グラフィックを楽しみたい人のために**

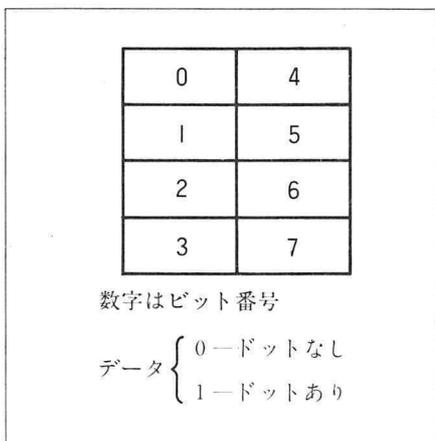
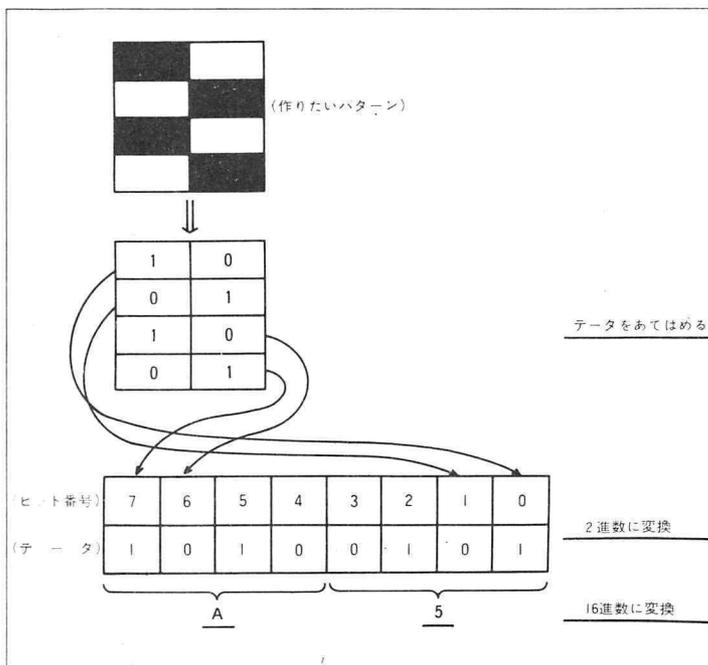
前項のことがわかれば，これは簡単です。

COLOR　7，0，1↗
PRINT　CHR$(12)↗

でカラー・モードにし，

LINE (X_1, Y_1) — (X_2, Y_2)
　　，PRESET，n，BF
　　　　　↑
　　　色の番号を指定

で画面の好きな位置を好きな色にセットします。あとは前項の方法にしたがえば良いのです。

以上のテクニックを使えば，BASICとマシン語により**カラー・グラフィックの高速ゲーム**が本書の知識内で可能となります。いろいろ実験してみてください。

《第4-5図》グラフィック・パターン変換図

《第4-6図》グラフィック・コードを作る

インベーダーの１列編隊

(注:侵略者という意味のキャラクター。以下同じ)

4. インベーダーへの挑戦

それではお待ちかね，

インベーダーの行進

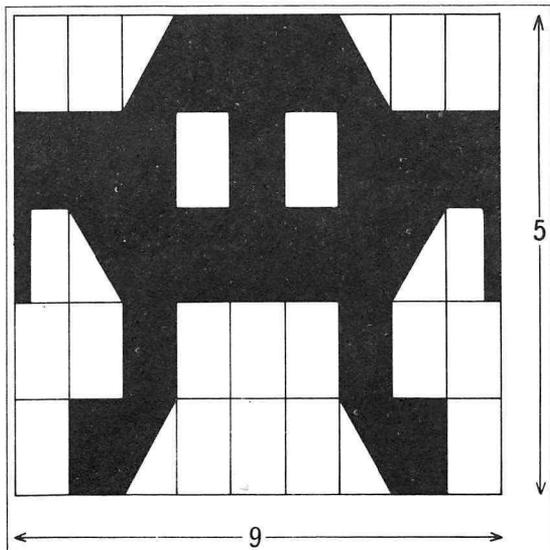
にむかって進みましょう。まずは，インベーダー１匹です。

> **〈チャレンジ７〉**
>
> インベーダーを（20，10）の位置に表示しなさい。

ここで（20，10）とは，

LOCATE　20，10

に相当する位置です。以下，このような記法を用いることにします。

まず**インベーダーのキャラクター**を作りましょう。**第4-7図**のように９×５の大きさに設定してみました。

《第４-７図》インベーダーのキャラクター設定

あなたの手で，別のキャラクターで設定してみてください。**第4-8図**が，〈チャレンジ７〉に対するプログラム例です。このD01F番地以降にインベーダーのキャラクターを格納してあります。

《第4-8図》インベーダーの表示

```
******************
* << ﾁｬﾚﾝｼﾞ 7 >> *
******************

                      ORG   0D000H
                 ;
F7C4             TEST:  EQU   0F7C4H          ;locate 20,10
5C66             MON:   EQU   5C66H
                 ;
D000 21C4F7      MAIN:  LD    HL,TEST
D003 CD09D0             CALL  PIV
D006 C3665C             JP    MON
                 ;
D009 111FD0      PIV:   LD    DE,DIV          ;PRINT IV AT HL
D00C 0605               LD    B,5             ;ﾀﾃ
D00E C5          P1:    PUSH  BC
D00F 0609               LD    B,9             ;ﾖｺ
D011 1A          P2:    LD    A,(DE)
D012 77                 LD    (HL),A
D013 13                 INC   DE
D014 23                 INC   HL
D015 10FA               DJNZ  P2
D017 016F00             LD    BC,120-9
D01A 09                 ADD   HL,BC
D01B C1                 POP   BC
D01C 10F0               DJNZ  P1
D01E C9                 RET
                 ;
D01F 2020E487    DIV:   DC    '         '
D023 8787E520
D027 20
D028 87878720           DC    '         '
D02C 87208787
D030 87
D031 88E68787           DC    '         '
D035 878787E7
D039 97
D03A 20208720           DC    '         '
D03E 20208720
D042 20
D043 2087E720           DC    '         '
D047 2020E687
D04B 20
                 ;
D04C                    END
```

次にインベーダーを表示する方法を考えましょう。**第4-8図**のプログラムでは，

PIV：D009-D01E

のサブルーチンで表示しています。これは，汎用性を持たせるためで，

HLレジスタ←──ビデオRAMの位置

を入れてCALLすると，そこにインベーダーを表示するように作ってあります。ですからこのHLの値を変えることによって，画面の自由な位置にインベーダーを表示することができます。メイン・プログラムの

D000-D008番地

を見るとそうなっているのがわかりますね。

5. ENDマークを使わず

それでは，そのPIVを考えてみましょう。

前に我々は，**"文字列表示ルーチン"** を作りました。これを5行分適用すれば，もちろんできます。ただしそのときは，1行のおわりごとに00HのENDマークをお忘れなく！　ここでは別の方法でアプローチしてみましょう。方針は，次のとおりです。

①DJNZを使って5行分のループを考える。

②その一つのループの中を，さらにDJNZで9回ループさせ1行の9キャラクター分を表示させる。

こういうの，前に画面消去のところでやりましたね。それでは，実際のプログラムでそのことを見てみましょう。

6. インベーダー1匹，イッチョアリー

D009番地

この時点では，すでにメイン・ルーチンでHLレジスタの設定（インベーダー左上のビデオRAMの座標）が行われています。DEレジスタにインベーダーの文字列（キャラクター・コード列）の先頭番地を設定してやります。

D00C番地およびD01C番地

```
LD     B, 5
  ⸾
DJNZ   P1
```

で外側のループを作っています。5は，5行分繰り返せという意味ですね。

D00E番地およびD01B番地

```
PUSH   BC
  ⸾
POP    BC
```

外側のループ・カウンタBの値を壊わさないよう，スタック領域に保護しています。

D00E番地およびD015番地

```
LD     B, 9
  ⸾
DJNZ   P2
```

1行9キャラクター分の表示のため，9回のループを作っています。

D011～D014番地

1行分の表示です。DEのところのキャラクター・コードを，Aレジスタを仲介にHLレジスタのところのビデオRAMに転送しています。

```
INC    DE
INC    HL
```

は，次のキャラクターを表示するために各ポインターの値を進めているのです。

D017～D01A番地

これは何をしているところかおわかりになりますか？

HL←──HL＋120-9

の計算をしています。HLの値を**次の行の左側にセットしなおしている**のですね。120を足すと，1段下に下がります（ビデオRAM80＋アトリビュート・エリア40）。さらに9キャラクター分を左に戻してやります。

なお，アセンブラを使うと

```
LD     BC, 120-9
```

のような書き方をしても，きちんと

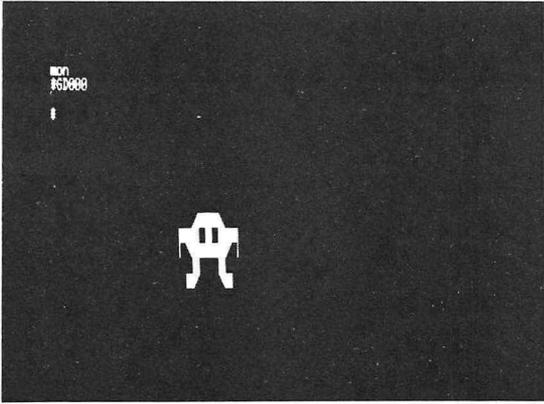
120-9＝111＝006FH

に変換してくれます。

さあ，これで〈チャレンジ7〉に対するプログラムが完成しました。**第4-8図**のプログラムを走らせてみてください。なお，プログラムを走らせる前に

```
WIDTH 80, 25
CONSOLE 0, 25
```

をお忘れなく！　**写真2**のように表示されます。

なお，このプログラムではインベーダーの表示をサブルーチン化していますから，HLの値をいろいろ変えると（ただし**ビデオRAMの範囲内におさまるように**），いろいろな位置に表示できます。また何回もCALLすれば，何匹でも表示できます。いろいろ試してみましょう。きっと**応用力がつきますよ。**

《写真２》第４－８図のプラグラムを実行

7. インベーダー1列編隊

次は，インベーダーの１列編隊に挑戦しましょう。もはや我々は，それを簡単に表示することができます。

〈チャレンジ8〉

（０，５）の位置から右に６匹のインベーダーを表示するプログラムを作りなさい。インベーダーとインベーダーの間は，１キャラクター分あけること。

先の〈チャレンジ７〉のサブルーチンを使えば，簡単にできそうですね。すなわちPIVを６回CALLすれば良いのです。ただし，１回CALLするたびにHLを10右に進めるのをお忘れなく。

8. サブルーチン化——汎用性

ここでは，またあとのことを考えて，６匹のインベーダーを表示するルーチンを，サブルーチン化してみました。すなわちHLレジスタに１番左のインベーダーの左上の座標（ビデオRAMの番地）を指定してCALLすると，そこから右に６匹のインベーダーが表示されるというものです。ここではそのサブルーチンにSRGと名前をつけてあります。

でき上がったプログラムが，**第4-9図**です。まずメイン・ルーチンから見ていきましょう。

《第4-9図》インベーダーの編隊

```
******************
* << ﾁｬﾚﾝｼﾞ 8 >> *
******************

                        ORG  0D000H
                   ;
F556               TEST:  EQU  0F556H      ;locate 0,5
5C66               MON:   EQU  5C66H
                   ;
D000 2156F5        MAIN:  LD   HL,TEST
D003 CD09D0               CALL SRG
D006 C3665C               JP   MON
                   ;
D009 0606          SRG:   LD   B,6
D00B C5            S:     PUSH BC
D00C E5                   PUSH HL
D00D CD19D0               CALL PIV
D010 E1                   POP  HL
D011 010A00               LD   BC,10
D014 09                   ADD  HL,BC
D015 C1                   POP  BC
D016 10F3                 DJNZ S
D018 C9                   RET
                   ;
D019 112FD0        PIV:   LD   DE,DIV      ;PRINT IV AT HL
D01C 0605                 LD   B,5         ;[illegible]
D01E C5            P1:    PUSH BC
D01F 0609                 LD   B,9         ;[illegible]
D021 1A            P2:    LD   A,(DE)
D022 77                   LD   (HL),A
D023 13                   INC  DE
D024 23                   INC  HL
D025 10FA                 DJNZ P2
D027 016F00               LD   BC,120-9
D02A 09                   ADD  HL,BC
D02B C1                   POP  BC
D02C 10F0                 DJNZ P1
D02E C9                   RET
                   ;
D02F 2020E487 DIV:        DC   '[illegible]'
D033 8787E520
D037 20
D038 87878720             DC   '[illegible]'
D03C 87208787
D040 87
D041 88E68787             DC   '[illegible]'
D045 878787E7
D049 97
D04A 20208720             DC   '[illegible]'
D04E 20208720
D052 20
D053 2087E720             DC   '[illegible]'
D057 2020E687
D05B 20
                   ;
D05C                      END
```

HLの位置を保護しておく

PIVの中でBが変化するので保護ループ

D000～D008番地

（０，５）をビデオRAMの番地に変換すると，

TEST＝F558H

になります。あとは，SRGをCALLするだけですね。

次がサブルーチンSRGです。

D009番地およびD015番地

６区分の表示のため，６回のループを作っています。

D00B番地およびD015番地

Bレジスタの値を保護しておかないと，PIVの中でBレジスタを使っていますから正しく６回ループされません。

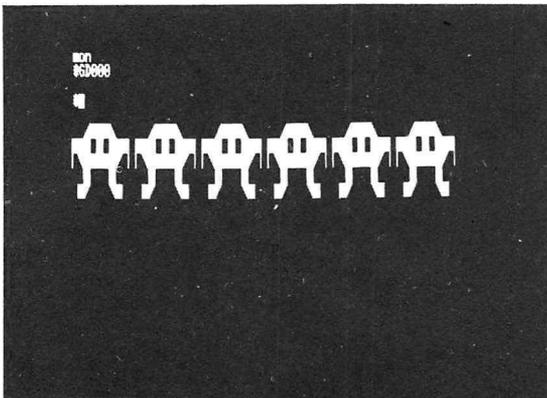
D00C番地およびD018番地

HLの値をPUSHしておけば，PIVをCALLしたあとでもPOPでもとに戻せますから，インベーダーの次の位置（10個右）を計算するのが楽です。

D011〜D014番地

HL←─HL＋10

の計算です。なぜこの計算をしているのかわかりますね（インベーダー９＋空白１）？

あとの部分は，〈チャレンジ７〉と同じものを使っていますから，これで〈チャレンジ８〉はおしまいです。さあ，プログラムを走らせましょう。**写真3**です。いかがですか？　少しは，マシン語に自信がつきましたか？

《写真３》チャレンジ８のプログラム実行

9. インベーダーのスパーク

さあ，さらに進みます。今度は，次のようなのはいかがでしょうか？

> **〈チャレンジ9〉**
>
> 〈チャレンジ８〉で表示したインベーダーの１列編隊をスパークさせなさい。

スパークというのは，マイコンでは

高速で表示を点滅させる

ことを言います。さあ，どう組みましょうか？

プログラムの流れは，次のようになります。

インベーダーの編隊を消す
タイマー1
インベーダーの編隊を表示する
タイマー2

ここで**タイマー**とは，

少し間をおく

ことを言います。**タイマーをおかないと点滅が早すぎてスパークに見えません。**なお，インベーダーが表示している時間を，インベーダーが消えている時間より長く取った方が自然に見えます。つまり

タイマー1　＜　タイマー2

にすると良いでしょう。

タイマーは，リアル・タイムGAMEを作るときの必需品ですから，これもサブルーチンにしておきましょう。作り方の原理は，

ムダなことをする命令

を何回も繰り返せば良いのです。回数を長くすれば長いタイマーが，また短くすれば短いタイマーが作れます。**第4-10図**のプログラム例では，

LD　　B，B

という命令を使いました。BレジスタにBレジスタの値を入れても，何の変化もありませんからムダですね？

10. スパーク・ルーチンの解析

それでは**第4-10図**のプログラム例を解析して行くことにします。

D000～D007番地

メイン・ルーチンです。全体で無限ループを作っているのがわかります。1回のループでサブルーチンSPRKをCALLしています。SPRKは，1回分のスパークを担当します。そのときD000のHLの値を変えてやれば，インベーダーの表示位置を変えることもできます。

次は，SPRKです。

D008～D00F番地

インベーダー1列編隊の消去です。といっても何のことはなく，**画面消去**してやれば全部消えてくれます。画面消去なら前に作りましたね。あれを利用すれば良いのですが，ここではキー・インの手間を考えて

ROM内の消去ルーチン

を利用しています。D009番地でCALLしている

CLR=045AH

がそれです。

その下のTIMEはタイマー・ルーチンです。

D010～D018番地

インベーダー1列編隊の表示です。〈チャレンジ8〉で作ったSRGをCALLしていますね。なおその下でTIMEを2回表示しているのは，消去の2倍の長さのタイマーを取るためです。

《第4-10図》インベーダーのスパーク

```
******************
* << ﾁｬﾚﾝｼﾞ 9 >> *
******************
                  ;
                        ORG   0D000H
                  ;
045A              CLR:  EQU   45AH          ;PRINT CHR$(12)
F558              TEST: EQU   0F558H        ;LOCATE 0,5
5C66              MON:  EQU   5C66H
                  ;
D000 2158F5       MAIN: LD    HL,TEST
D003 CD08D0             CALL  SPRK
D006 18F8               JR    MAIN
                  ;
D008 E5           SPRK: PUSH  HL        ┐
D009 CD5A04             CALL  CLR       │ 消去
D00C CD40D0             CALL  TIME      │
D00F E1                 POP   HL        ┘
D010 CD1AD0             CALL  SRG       ┐
D013 CD40D0             CALL  TIME      │ 表示
D016 CD40D0             CALL  TIME      ┘
D019 C9                 RET
                  ;
D01A 0606         SRG:  LD    B,6           ;IV=6
D01C C5           S:    PUSH  BC
D01D E5                 PUSH  HL
D01E CD2AD0             CALL  FIV
D021 E1                 POP   HL
D022 010A00             LD    BC,10         ;ﾐｷﾞﾍ
D025 09                 ADD   HL,BC
D026 C1                 POP   BC
D027 10F3               DJNZ  S
D029 C9                 RET
                  ;
D02A 114CD0       FIV:  LD    DE,DIV        ;PRINT IV AT HL
D02D 0605               LD    B,5
D02F C5           F1:   PUSH  BC
D030 0609               LD    B,9           ;ﾖｺ
D032 1A           F2:   LD    A,(DE)
D033 77                 LD    (HL),A
D034 13                 INC   DE
D035 23                 INC   HL
D036 10FA               DJNZ  F2
D038 016F00             LD    BC,120-9
D03B 09                 ADD   HL,BC
D03C C1                 POP   BC
D03D 10F0               DJNZ  F1
D03F C9                 RET
                  ;
D040 06FF         TIME: LD    B,0FFH    ── 外側のループ
D042 C5           T1:   PUSH  BC        ── 外側のループ・カウンタの保護
D043 06FF               LD    B,0FFH    ── 内側のループ
D045 40           T2:   LD    B,B
D046 10FD               DJNZ  T2
D048 C1                 POP   BC
D049 10F7               DJNZ  T1
D04B C9                 RET
                  ;
D04C 2020E487     DIV:  DC    '         '
D050 8787E520
D054 20
D055 87878720           DC    '         '
D059 87208787
D05D 87
D05E 88E68787           DC    '         '
D062 878787E7
D066 97
D067 20208720           DC    '         '
D06B 20208720
D06F 20
D070 2087E720           DC    '         '
D074 2020E687
D078 20

D079                    END
```

SRG，PIVについては，前に解析したのと同じルーチンを使っていますから，あとはTIMEを解析しておしまいです。

TIMEの本体は，前述したように

LD　　B，B

という意味のない命令です。これをDJNZにより繰り返しています。しかも1回のループでは短かすぎますので(ここがマシン語の高速性の現われているところ)，**2重ループで構成**しています。

D040番地とD049番地

外側のループを作っています。ループ・カウンターのBレジスタには，最高の

FFH＝255

を設定しています。

D042番地とD048番地

外側のループ・カウンターBレジスタの保護です。もう慣れましたね。

D043番地とD046番地

中側のループです。ここでも最高の

FFH＝255

を設定しています。

以上で〈チャレンジ9〉のプログラムは，おしまいです。さっそく走らせてみてください。インベーダーの点滅が見られるでしょう。

このスパークの手法がわかると，**いろいろな応用**ができます。たとえばタイマーを短かくすれば，

インベーダーがやられたとき

GAME OVER　のとき

等に使えそうです。また，1回スパークするたびに，インベーダーの表示位置を変えるとどうなるでしょうか？　何と

インベーダーが動き出す！

ではありませんか。それを取り入れたのが，次の〈チャレンジ10〉です。

11. インベーダーの移動

いよいよ本書の**最後のチャレンジ**となりました。ここまで読破されてきたあなたに敬意を表すると同時に最後まで気を抜かず読み続けてください。ゴールはま近です。

> **〈チャレンジ10〉**
>
> インベーダーの1列編隊を，6行目の左端から右端まで行進させるプログラムを作りなさい。

それでは最後のプログラム例を一緒に見ていくことに致しましょう。**第4-11図**になります。

まずメイン・ルーチンです。

D000-D008番地

HLレジスタにインベーダーのスタート位置を入れて，サブルーチンGOをCALLするだけです。HLの値を変えれば，どの行でもインベーダーを行進させることができます。

そして，いよいよ最後のサブルーチンの解析となりました。インベーダーの移動を司るGO（D009～D015番地）です。

① 全体は，DJNZによる一つのループを作っています。D009番地でループ回数を21回に指定していますが，これについては**第4-12図**をみてください。画面サイズが横80，そしてインベーダーの編隊の大きさが59ですから，21キャラクター分を右に移動すれば右端に着くことになります。

② 1回の移動処理は，サブルーチンSPRKで行います。インベーダーの位置HLは，スタック領域に保存しておきましょう。

その他のルーチンは，全て既出のものを使っています。ただしタイマーは1/4の長さにしています(D04E番地を見てください)。

それでは，本書最後のプログラムを走らせてみましょう。**画面モードを80×25**にしておくのをお忘れなく！　プログラム・スタートは，

GD000↗

です。**写真4**にインベーダーの移動途中，また**写真5**に

《第4-11図》インベーダーの行進

```
+++++++++++++++++++
+ << ﾁｬﾚﾝｼﾞ 10 >> +
+++++++++++++++++++
                    ;
                           ORG   0D000H
                    ;
045A                CLR:   EQU   45AH           ;PRINT CHR$(12)
F558                TEST:  EQU   0F558H         ;LOCATE 0,5
5C66                MON:   EQU   5C66H
                    ;
D000 2158F5         MAIN:  LD    HL,TEST
D003 CD09D0                CALL  GO
D006 C3665C                JP    MON
                    ;
D009 0615           GO:    LD    B,21
D00B C5             G1:    PUSH  BC
D00C E5                    PUSH  HL
D00D CD16D0                CALL  SPRK
D010 E1                    POP   HL
D011 23                    INC   HL
D012 C1                    POP   BC
D013 10F6                  DJNZ  G1
D015 C9                    RET
                    ;
D016 E5             SPRK:  PUSH  HL
D017 CD5A04                CALL  CLR
D01A CD4ED0                CALL  TIME
D01D E1                    POP   HL
D01E CD28D0                CALL  SRG
D021 CD4ED0                CALL  TIME
D024 CD4ED0                CALL  TIME
D027 C9                    RET
                    ;
D028 0606           SRG:   LD    B,6            ;IV=6
D02A C5             S:     PUSH  BC
D02B E5                    PUSH  HL
D02C CD38D0                CALL  PIV
D02F E1                    POP   HL
D030 010A00                LD    BC,10          ;ﾐｷﾞﾍ
D033 09                    ADD   HL,BC
D034 C1                    POP   BC
D035 10F3                  DJNZ  S
D037 C9                    RET
                    ;
D038 115AD0         PIV:   LD    DE,DIV         ;PRINT IV AT HL
D03B 0605                  LD    B,5
D03D C5             P1:    PUSH  BC
D03E 0609                  LD    B,9            ;ﾖｺ
D040 1A             P2:    LD    A,(DE)
D041 77                    LD    (HL),A
D042 13                    INC   DE
D043 23                    INC   HL
D044 10FA                  DJNZ  P2
D046 016F00                LD    BC,120-9
D049 09                    ADD   HL,BC
D04A C1                    POP   BC
D04B 10F0                  DJNZ  P1
D04D C9                    RET

D04E 0640           TIME:  LD    B,40H
D050 C5             T1:    PUSH  BC
D051 06FF                  LD    B,0FFH
D053 40             T2:    LD    B,B
D054 10FD                  DJNZ  T2
D056 C1                    POP   BC
D057 10F7                  DJNZ  T1
D059 C9                    RET
                    ;
D05A 2020E487       DIV:   DC    '         '
D05E 8787E520
D062 20
D063 87878720              DC    '         '
D067 87208787
D06B 87
D06C 88E68787              DC    '         '
D070 878787E7
D074 97
D075 20208720              DC    '         '
D079 20208720
D07D 20
D07E 2087E720              DC    '         '
D082 2020E687
D086 20
                    ;
D087                       END
```

移動距離は21
ループ・カウンタの保護
インベーダーの位置保護

《第4-12図》インベーダーの移動距離は？

移動終了時の様子を示しておきます。

<第4ブロックのおわりに>

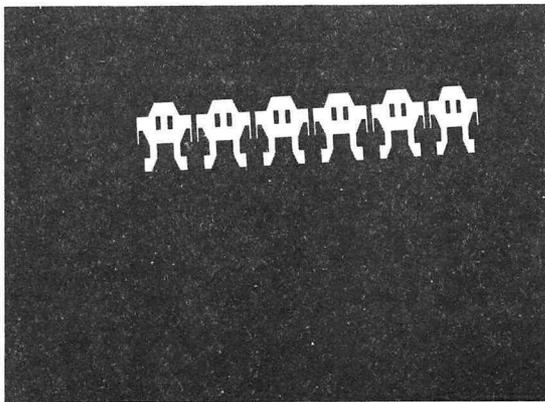
とうとう最後のブロックも，終りに到達したようです。**あなたとの共同作業**も，ひとまずここでお別れすることになります。もう私からは何も申し上げることはありません。また紙面の余裕もありません。ただ一つだけアドバイスしておきます。あなたのこの1ページに**今日の日付**を書き込んでおいてください。それは，あなたのマイコン・ライフにとって記念すべき日です。それは，あなたが初めてマシン語の解説書1冊を

読破した

日です。

<本日の日付>

年　　月　　日（　　）

サイン

《写真4》インベーダーの移動途中

《写真5》インベーダーの移動終了

付録

マシン語を使うために

アプリケーション・プログラム

「逆アセンブラ」

はじめに

ＰＣ－8001は，パーソナル・コンピュータであり，強力なＮ－ＢＡＳＩＣがROMで供給されているため，スイッチ・オンでプログラムを楽しむことができます。また同時に，モニタ・プログラムも内蔵されていて，ＢＡＳＩＣから簡単に呼び出せますから，マシン語のプログラムも出来るわけです。

このモニタ・プログラムには**ダンプ機構**が付いていますから，マシン語プログラムを**ヘキサ**（16進数）の形で見ることは出来ます。たとえば

Ｄ000　Ｃ3　66　5Ｃ────①

のように。これは自分の作ったプログラムでも，他人様の作ったプログラムでも，ＲＯＭ内のプログラムでも同様です。

しかし①のような16進数の羅列では，我々人間にピンときません。そこで「Ｚ－80命令表」を片手に①の16進数を，

ＪＰ　5Ｃ66Ｈ

と変換することになります。この作業，プログラムが短いうちは良いのですが，長いプログラムになると単純作業ゆえ，馬鹿馬鹿しくなってきます。とくに，他人様のお作りになったプログラムを解析しようとするときは，解析の前に，この長い変換作業が必要となってくるのです。

そこでその作業を，マイコンにやらせてしまおうという考えが起こってきます。そのような必要性から生まれてきたのが，

逆アセンブラ

です。

本「逆アセンブラ」の特色

ＰＣ－8001用の「逆アセンブラ」は，種々のものが発表されています。市販のものでも10,000円を越えるものまで，いろいろ販売されています。それぞれ良く出来ていると思います。しかし，自分のプリンタに合わなかったりして，自分にピッタリのものがなかなか見つかりませんでした。やはり自分で作るのが，自分好みにするには一番です。

私の「逆アセンブラ」の特色は次の通りです。

① 1列モード，2列モードが選択出来る。

印字見本をごらんください。**第1図**はＰＣ－8001のＲＯＭ内5Ｃ99番地～5Ｄ67番地を2列モードで逆アセンブルしたものです。余白の部分は少なくても良いから，**プリンタ用紙を2倍に使いたいとき**に使用してください。なお，この番地にはモニタのＳコマンド，およびＤコマンドが書かれていますので，解読すると面白いと思います。

第2図は5Ｃ2Ｃ番地から5Ｃ92番地までを，1列モードで印字したものです。右側の部分にかなりの余白が出来ますから，いろいろ書き込みしたい時等に使います。

なお，1列モード，2列モードの併用も可能です。

② プログラム・エリアとデータ・エリアの分離可能。

これは重要な問題ですので，一応説明しておきます。今**第3図**にあるマシン語のダンプ・リストを逆アセンブルしたとします。**第4図**がそうです。これを解読してみてください。Ｄ00Ｆ番地あたりからムチャクチャに逆アセンブルしています。

正しく逆アセンブルされていない理由は，

```
5C99 AF          XOR  A
5C9A 3238FF      LD   (0FF38H),A
5C9D 67          LD   H,A
5C9E 6F          LD   L,A
5C9F CDAD5F      CALL 5FADH
5CA2 FE0D        CP   0DH
5CA4 2861        JR   Z,5D07H
5CA6 CD395E      CALL 5E39H
5CA9 D8          RET  C
5CAA CD4B5E      CALL 5E4BH
5CAD CD215E      CALL 5E21H
5CB0 C0          RET  NZ
5CB1 CDCA5F      CALL 5FCAH
5CB4 CDC05E      CALL 5EC0H
5CB7 CDD45F      CALL 5FD4H
5CBA CDBD5E      CALL 5EBDH
5CBD 3E2D        LD   A,2DH
5CBF CD5702      CALL 0257H
5CC2 CDE20B      CALL 0BE2H
5CC5 CDB95F      CALL 5FB9H
5CC8 FE0D        CP   0DH
5CCA 2840        JR   Z,5D0CH
5CCC FE20        CP   20H
5CCE 282C        JR   Z,5CFCH
5CD0 FE08        CP   08H
5CD2 2830        JR   Z,5D04H
5CD4 CD5702      CALL 0257H
5CD7 F5          PUSH AF
5CD8 CDE20B      CALL 0BE2H
5CDB F1          POP  AF
5CDC CD395E      CALL 5E39H
5CDF 3831        JR   C,5D12H
5CE1 57          LD   D,A
5CE2 CDAD5F      CALL 5FADH
5CE5 CD395E      CALL 5E39H
5CE8 3828        JR   C,5D12H
5CEA 5F          LD   E,A
5CEB CDA05E      CALL 5EA0H
5CEE 77          LD   (HL),A
5CEF 2C          INC  L
5CF0 2004        JR   NZ,5CF6H
5CF2 24          INC  H
5CF3 2817        JR   Z,5D0CH
5CF5 25          DEC  H
5CF6 2D          DEC  L
5CF7 CD5A5E      CALL 5E5AH
5CFA 18BB        JR   5CB7H
5CFC CDD45F      CALL 5FD4H
```

```
5CFF CDD45F      CALL 5FD4H
5D02 18EB        JR   5CEFH
5D04 2B          DEC  HL
5D05 18AA        JR   5CB1H
5D07 2A30FF      LD   HL,(0FF30H)
5D0A 18A5        JR   5CB1H
5D0C 2230FF      LD   (0FF30H),HL
5D0F C3665C      JP   5C66H
5D12 2230FF      LD   (0FF30H),HL
5D15 C9          RET
5D16 3EFF        LD   A,0FFH
5D18 3238FF      LD   (0FF38H),A
5D1B 210000      LD   HL,0000H
5D1E 54          LD   D,H
5D1F 5C          LD   E,H
5D20 CD215E      CALL 5E21H
5D23 2037        JR   NZ,5D5CH
5D25 54          LD   D,H
5D26 5D          LD   E,L
5D27 C5          PUSH BC
5D28 010F00      LD   BC,000FH
5D2B 09          ADD  HL,BC
5D2C EB          EX   DE,HL
5D2D C1          POP  BC
5D2E CDD20B      CALL 0BD2H
5D31 CDCA5F      CALL 5FCAH
5D34 CDC05E      CALL 5EC0H
5D37 CDD45F      CALL 5FD4H
5D3A CDBD5E      CALL 5EBDH
5D3D CDF10C      CALL 0CF1H
5D40 DA665C      JP   C,5C66H
5D43 FE13        CP   13H
5D45 200A        JR   NZ,5D51H
5D47 CDF10C      CALL 0CF1H
5D4A DA665C      JP   C,5C66H
5D4D FE13        CP   13H
5D4F 20F6        JR   NZ,5D47H
5D51 CDD35E      CALL 5ED3H
5D54 CA665C      JP   Z,5C66H
5D57 CD5A5E      CALL 5E5AH
5D5A 18DB        JR   5D37H
5D5C EB          EX   DE,HL
5D5D CD215E      CALL 5E21H
5D60 C0          RET  NZ
5D61 CDD35E      CALL 5ED3H
5D64 D8          RET  C
5D65 EB          EX   DE,HL
5D66 18C6        JR   5D2EH
```

《第１図》逆アセンブラ２列印字見本（ROM内ルーチンのSコマンド，Dコマンドを逆アセンブルしたもの）

　　プログラムの途中にある

　　データ領域

のためで，逆アセンブラがＤ００Ｆ番地から始まっている**データ領域を，プログラムと勘違いして逆アセンブルにしてしまったから**です。

第５図が，プログラム領域とデータ領域を分離して，正しく逆アセンブルしたものです。これなら十分に解読できるでしょう。しかし，プログラム領域とデータ領域を分けることの出来ない逆アセンブラでは，**第３図**のプログラムを逆アセンブルすることは不可能です。もっとも，**プログラム領域とデータ領域の指定はあなたの仕事ですが。**

なお，このプログラムを

　　　ＧＤ０００↗

で走らせると，

　　デンパシンブン：マイコン

と表示されるのはお分かりですね？

以上が私の逆アセンブラの特色です。

```
5C2C E3          DI
5C2D 3AFF7F      LD   A,(7FFFH)
5C30 FE55        CP   55H
5C32 CAFC7F      JP   Z,7FFCH
5C35 2234FF      LD   (0FF34H),HL
5C38 ED7336FF    LD   (0FF36H),SP
5C3C 015E5C      LD   BC,5C5EH
5C3F C5          PUSH BC
5C40 3E2A        LD   A,2AH
5C42 CD5702      CALL 0257H
5C45 CDE20B      CALL 0BE2H
5C48 CDAD5F      CALL 5FADH
5C4B 21875C      LD   HL,5C87H
5C4E 010C00      LD   BC,000CH
5C51 EDB1        CPIR
5C53 C0          RET  NZ
5C54 216F5C      LD   HL,5C6FH
5C57 09          ADD  HL,BC
5C58 09          ADD  HL,BC
5C59 5E          LD   E,(HL)
5C5A 23          INC  HL
5C5B 56          LD   D,(HL)
5C5C EB          EX   DE,HL
5C5D E9          JP   (HL)
5C5E CDCA5F      CALL 5FCAH
5C61 3E3F        LD   A,3FH
5C63 CD5702      CALL 0257H
5C66 ED7B36FF    LD   SP,(0FF36H)
5C6A CDCA5F      CALL 5FCAH
5C6D 18CD        JR   5C3CH
5C6F 665C935C    DB   66H,5CH,93H,5CH
5C73 665C665C    DB   66H,5CH,66H,5CH
5C77 665C3C5C    DB   66H,5CH,3CH,5CH
5C7B E65D745D    DB   0E6H,5DH,74H,5DH
5C7F AE5D685D    DB   0AEH,5DH,68H,5DH
5C83 165D995C    DB   16H,5DH,99H,5CH
5C87 5344474C    DB   53H,44H,47H,4CH
5C8B 57540C0D    DB   57H,54H,0CH,0DH
5C8F 0A20021B    DB   0AH,20H,02H,1BH
```

《第2図》逆アセンブラ1列印字の見本（モニタのスタートから，ジャンプ・テーブルまでを逆アセンブルしたもの。データ・エリアとプログラム・エリアを分けて印字できる）

プログラムの走らせ方

プログラムはオールＢＡＳＩＣですので，フリー・エリアの前の方に格納されます。マシン語プログラムは，普通フリー・エリアの後部に置かれますので，逆アセンブラとしてはＢＡＳＩＣで十分です。事実，自分でこのプログラムを愛用していますが，マシン語と重なって困ったことはありません。

リストのプログラムを入力したら，逆アセンブルをしたいプログラムをロードするわけですが，その前に

ＣＬＥＡＲ　３００，＆Ｈ××××
↑マシン語プログラムのスタート番地－１

を行なって，マシン語プログラムを保護してください。

なお，「逆アセンブラ」とマシン語プログラムのロードは，どちらが先でもかまいません。

以上，２本のプログラムを入力したら，

ＲＵＮ↗

プログラムはスタートします。

D000	CD	09	D0	CD	1B	D0	C3	66
08	5C	21	0F	D0	C3	ED	52	C3
10	DE	DD	CA	DF	BC	DD	3C	DE
18	DD	3A	00	21	21	D0	C3	ED
20	52	CF	B2	BA	DD	00		

《第３図》ダンプ・リスト（これを逆アセンブルしてみる）

```
D000 CD09D0      CALL 0D009H
D003 CD1BD0      CALL 0D01BH
D006 C3665C      JP   5C66H
D009 210FD0      LD   HL,0D00FH
D00C C3ED52      JP   52EDH
D00F C3DEDD      JP   0DDDEH
D012 CADFBC      JP   Z,0BCDFH
D015 DDCC
D017 DEDD        SBC  A,0DDH
D019 3A0021      LD   A,(2100H)
D01C 21D0C3      LD   HL,0C3D0H
D01F ED52        SBC  HL,DE
D021 CF          RST  08H
D022 B2          OR   D
D023 BA          CP   D
D024 DD00
```

「DD CC」はZ-80にない(未定義命令)←のため逆アセンブルされない

（D017～D024）間違ったプログラムで解読不能になっている

《第４図》単純に逆アセンブルする（第３図のダンプリストを単純に逆アセンブルしたもの。明らかに誤り）

```
D000 CD09D0     CALL 0D009H    ┐
D003 CD1BD0     CALL 0D01BH    │
D006 C3665C     JP   5C66H     │ プログラム領域
D009 210FD0     LD   HL,0D00FH │
D00C C3ED52     JP   52EDH     ┘
D00F C3DEDDCA   DB   0C3H,0DEH,0DDH,0CAH ┐
D013 DFBCDDCC   DB   0DFH,0BCH,0DDH,0CCH │ データ領域
D017 DEDD3A00   DB   0DEH,0DDH,3AH,00H   ┘
D01B 2121D0     LD   HL,0D021H ┐ プログラム領域
D01E C3ED52     JP   52EDH     ┘
D021 CFB2BADD   DB   0CFH,0B2H,0BAH,0DDH ┐ データ領域
D025 00         DB   00H                 ┘
```

《第5図》正しく逆アセンブルする（プログラム・エリアとデータ・エリアを分離できない逆アセンブラでは，このようにアセンブルすることは不可能）

使い方

写真1が，プログラム・スタート直後を表わしています。まず，プリンタの指定です。使わないときは，“0”を，使うときは“1”を指定してください。

続いて「スタート・アドレス」と「エンド・アドレス」を聞いてきます。0～4ケタの16進数で答えてください。何も入力しないでRETキーを押すと，0番地とみなされます。なおここで

スタート・アドレス＞エンド・アドレス

のように不合理な入力をすると，何度も聞き返してきます。

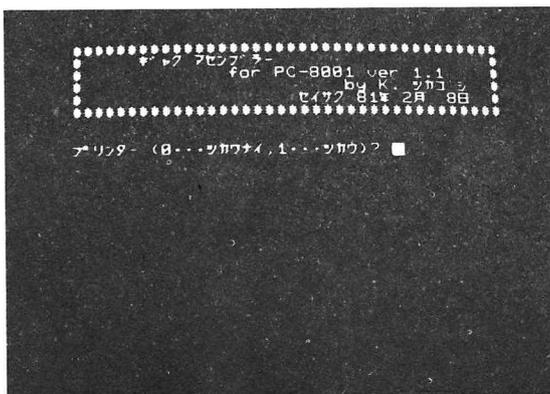
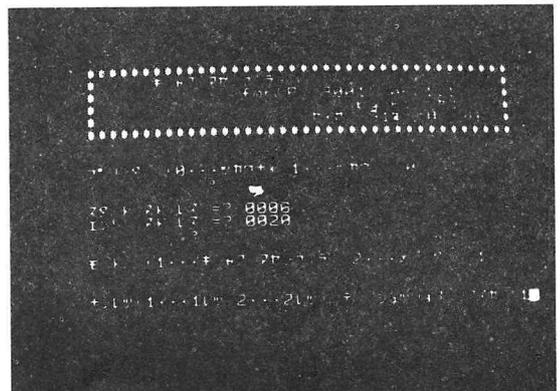
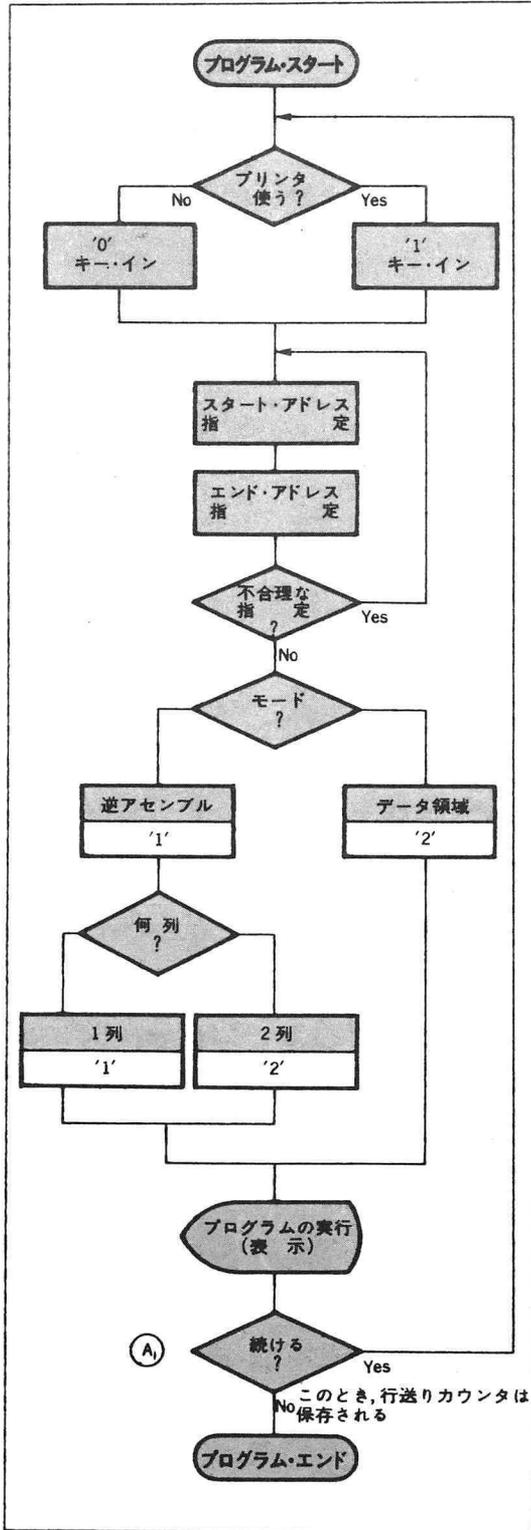
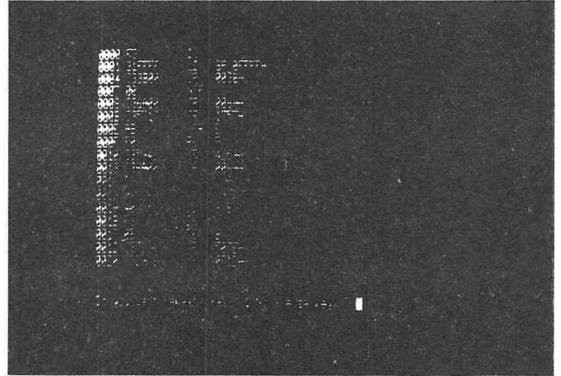
次に本逆アセンブラの特色である「モード」を指定します。逆アセンブルするときは“1”を，データ領域のときは“2”をキー・インしてください。

次にその指定が“1”の逆アセンブル・モードのときは，何列で印字するか聞いてきます。1列のときは“1”を，また経済的に2列で印字するときは“2”を指定してください。なお，「データ」の指定のときは，直ちに実行されます。というよりは，データ領域に限っては1列モードしか使えません。

写真2は，「プリンタは使わず(0)，0番地から20番地まで逆アセンブル・モード(1)で，1列」を指定したところです。最後にRETキーを押して実行したのが，写真3です。

プログラムの実行が終わると，さらに続けるかを聞いてきます。この機能は重要ですので，あとでもう一度触れます。続けるときは“1”を，やめるときは，“1”以外の数を入力してください。

以上の手続きを図解したのが，第6図です。

ｷﾞｬｸ ｱｾﾝﾌﾞﾗｰ
for PC-8001 ver 1.1
by K. ﾂｶｺﾞｼ
ｾｲｻｸ 81ﾈﾝ 2ｶﾞﾂ 8ﾆﾁ
ﾌﾟﾘﾝﾀｰ (0･･･ﾂｶﾜﾅｲ,1･･･ﾂｶｳ)?

《写真1》逆アセンブラ初期画面

《写真2》逆アセンブラの条件入力

《第6図》「逆アセンブラ」使用手続一覧

《写真3》逆アセンブラ実行

プリンタ・モードの注意

プリンタを使用する場合の注意点をまとめておきます。

① プリンタ用紙の使い方は，**第7図**の通りです。
まず1枚の用紙には62字印字すると改行します。これらはプログラム内部に行のカウンタを設けているらです。したがって最初にプリンタ用紙をセットするときに，図のように上から3行目から印字を開始するようにセットしておくとよいでしょう。そうすれば，1枚の用紙の上下2行ずつが空くようになって見やすいと思います。

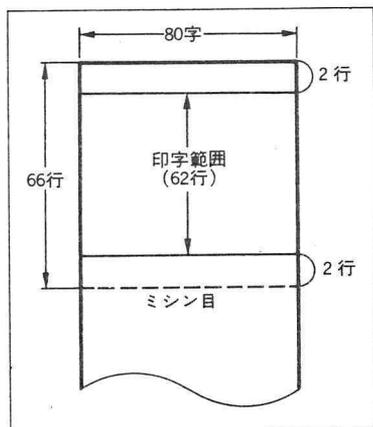
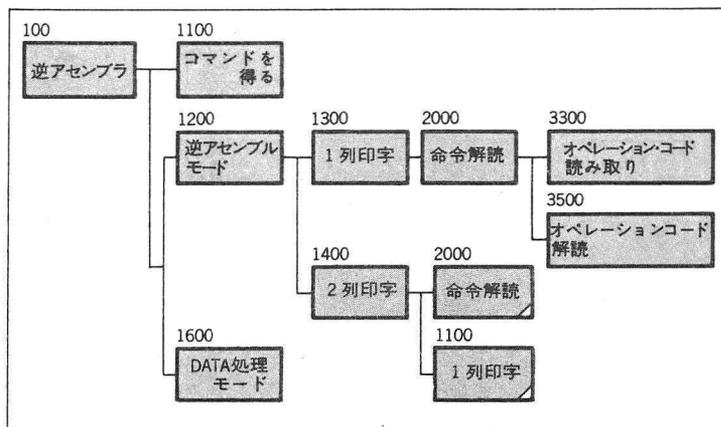
② 経済用の2列モードについても，62行毎に改行するようにセットしてあります。これは2列モードに限って**2パス方式**を採用していて，1パス目で右の列の先頭アドレスを計算するというぜいたくなことをしているからです。

③ 印字モードについては，印字と同時にCRTにも表示するようにしました。なお，ページがえのときはCRT上では2行復改するようにしました。このためCRTを見ているだけで，プリンタの監視が可能です。

④ 逆アセンブル・モードとデータ領域の印字を併用するとき，各モード終了時の

Do you want repeat ?

のメッセージ(**第6図**のⒶのところ)で，**必ず"1"を指定**してください。こうすることにより，行のカウンタが保存されますから，以後もその紙の62行目で改行されます。

以上の点に注意していただければ，美しい逆アセンブル・リストが取れると思います。

《第７図》プリンタ用紙の使われ方

《第８図》全体構造図

プログラムについて

① 先にも述べましたようにＰＣ－8001用の逆アセンブラは数種類発表・発売されています。そのうちのいくつかを，私も実際に使用してみました。それぞれ良く出来ているのですが，やはり自分の好みに改良したくなります。どうせ改良するなら自分で作ってしまおうと思って作ったのが，本プログラムです。市販の逆アセンブラは結構高い値段がついています。リストをキー・インするのは大変ですが，もし時間があれば入力してみてください。無料で「逆アセンブラ」が手に入ります。

② プログラムはＢＡＳＩＣの範囲内で可能な限り，ストラクチャード・プログラミングを採用しました。そのため省メモリよりは，**基本構造を守る**ことに重点を置いています。

③ プログラムの解読にあたっては，
- 変数表
- 全体構造図（第8図）
- 重要モジュールＮＳチャート

を用意しましたので，活用してください。

④ 全体の流れを説明しておくと，100～130行がメイン・ルーチンです。

110行が変数の初期設定で，配列の準備や，ライン・カウンターＬ１を初期化しています。120行はファンクション・キーの設定ですから不要です。130行で画面モードをセットして，1100行の下位モジュールをCall しています。1100行からのルーチンでは各種コマンドを入力し，処理エンド・フラグをリセットして戻ってきます。

次いで処理モード変数ＧＤにより1200行からの逆アセンブラ処理か，1600行のＤＡＴＡ処理に分岐します。130行は処理を希望する限りループを続けます。

《第１表》変　数　表

PR	プリンタ・スイッチ｛0：プリンタ使わない／1：　〃　使う
AD	スタート・アドレス
A9	エンド・アドレス
GD	処理モード｛1：逆アセンブラ・モード／2：データ・モード
G$	解読された1行分が入る
LI	ライン・カウンター（1～62） 1ページ62行印字
OV	処理エンド・フラグ｛0：まだ処理が終わっていない／1：処理終了
AD$	解読した「アドレス＋オペレーション・コード」が入る

番外編のおわりに

最近はプリンタも安くなってきて，グラフィック印字まで可能な機種が出まわっています。私のプリンタは，残念ながらグラフィック印字は出来ません。

しかも高～～い値段で購入してガックリ来ている一人です。

《ＮＳチャート１》メイン・ルーチン

《ＮＳチャート３》１列印字

《ＮＳチャート２》コマンドを得る

《ＮＳチャート４》命令解読

ところが世の中広いもので，
「プリンタを買ってはみたが，ＢＡＳＩＣのリストを取るだけ」
というぜいたくなケースもあるようです。それではせっかくのプリンタがもったいないですね。

今回の「逆のアセンブラ」は，**プリンタ・ユーティリティ**の一つです。もしあなたのプリンタが，ＢＡＳＩＣのリスト取り専用でしたら，ぜひ使ってみてください。私もＰＣ－8001購入後，いろいろな種類のプログラムを作りましたが，この「逆アセンブラ」は最も愛用しているプログラムの一つです。

```
0 REM ****************************************
0 REM *       ｷﾞｬｸ ｱｾﾝﾌﾞﾗｰ                      *
0 REM *                 for PC-8001 ver 1.1    *
0 REM *                          by K. ﾂｶｺﾞｼ   *
0 REM *          ｾｲｻｸ   Ver 1.0  81年 2月  7日  *
0 REM *                 Ver 1.1       2月  8日  *
0 REM ****************************************
100 REM Main ﾙｰﾁﾝ
110 DEFINT B-Z:DIM A$(79),B$(3),C$(7),D$(7):FOR I=0 TO 79:READ A$(I):NEXT:FOR I=
0 TO 3:READ B$(I):NEXT:FOR I=0 TO 7:READ C$(I):NEXT:FOR I=0 TO 7:READ D$(I):NEXT
:LI=1
120 KEY 1,"motor"+CHR$(13):KEY 6,"width80,25:cons":KEY 7,"1E0,25,1,0"+CHR$(13):K
EY 9,"csave "+CHR$(34):KEY 10,"cload":POKE &HEADB,&H6F
130 WIDTH 40,25:CONSOLE 0,25,0,0:GOSUB 1110:OV=0:ON GD GOSUB 1210,1610:PRINT:PRI
NT:I=1:INPUT "Do you want repeat (Yes･･･1,No･･･else key) ";I:IF I=1 THEN 130 ELS
E END
1100 REM ﾀｲﾄﾙ ﾋｮｳｼﾞ & Command ｦ ｲﾚﾙ (>PR,AD,A9,GD)
1110 PRINT CHR$(12);:PRINT "****************************************":PRINT "*
 ｷﾞｬｸ ｱｾﾝﾌﾞﾗｰ                        *":PRINT "*                for PC-8001 ver 1.1    *
":PRINT "*                            by K. ﾂｶｺﾞｼ  *"
1120 PRINT "*                          ｾｲｻｸ 81年 2月  8日  *":PRINT "*********************
******************":PRINT:PRINT
1130 PR=0:INPUT "ﾌﾟﾘﾝﾀｰ (0･･･ﾂｶﾜﾅｲ;1･･･ﾂｶｳ)";PR:IF PR<>1 AND PR<>0 THEN 1130
1140 PRINT:PRINT:INPUT "ｽﾀｰﾄ ｱﾄﾞﾚｽ =";I$:AD=VAL("&H"+I$):IF AD<0 THEN AD=AD+6553
6!
1150 INPUT "ｴﾝﾄﾞ ｱﾄﾞﾚｽ =";I$:A9=VAL("&H"+I$):IF A9<0 THEN A9=A9+65536!
1160 IF A9<AD THEN 1150 ELSE PRINT:PRINT
1170 GD=1:INPUT "ﾓｰﾄﾞ (1･･･ｷﾞｬｸ ｱｾﾝﾌﾞﾗｰ;2･･･ﾃﾞｰﾀ)";GD:IF GD<>1 AND GD<>2 THEN 11
70 ELSE PRINT:PRINT:RETURN
1200 REM ｷﾞｬｸ ｱｾﾝﾌﾞﾗｰ
1210 I=1:INPUT "ﾅﾝﾚﾂ(1･･･1ﾚﾂ;2･･･2ﾚﾂ) ﾃﾞ ｼｭﾂﾘｮｸ ｼﾏｽｶ";I:IF I<1 OR I>2 THEN 1210
1220 PRINT CHR$(12);:WIDTH 80:ON I GOSUB 1310,1410:RETURN
1300 REM PRINT only LEFT (AD<start adress:OV=1)
1310 GOSUB 2010:PRINT G$:IF PR=1 THEN LPRINT G$
1320 LI=LI+1:IF LI=63 THEN LI=1:PRINT:PRINT:IF PR=1 THEN LPRINT CHR$(12);
1330 IF OV=0 THEN 1310 ELSE RETURN
1400 REM PRINT LEFT & RIGHT(AD<ﾍﾟｰｼﾞ ﾉ start adress)
1410 A1=AD:FOR I1=1 TO 62:GOSUB 2010:NEXT:A2=AD:AD=A1:OV=0:IF A2>A9 THEN GOSUB 1
310 ELSE GOSUB 1510
1430 IF OV=0 THEN 1410 ELSE RETURN
1500 REM LEFT & RIGHT ｻﾌﾞﾙｰﾁﾝ
1510 AD=A1:GOSUB 2010:A1=AD:PRINT G$;:IF PR=1 THEN LPRINT G$;
1520 IF A2>A9 THEN PRINT:IF PR=1 THEN LPRINT:GOTO 1540 ELSE 1540
1530 AD=A2:GOSUB 2010:A2=AD:PRINT TAB(40);G$:IF PR=1 THEN LPRINT TAB(40);G$
1540 LI=LI+1:IF LI<63 THEN 1510 ELSE LI=1:AD=A2:PRINT:PRINT:IF PR=1 THEN LPRINT
CHR$(12);
1590 RETURN
1600 REM DATA ｼｮﾘ
1610 PRINT CHR$(12);:WIDTH80
1620 G$="":GOSUB 4610:AD$=RIGHT$("000"+HEX$(AD-1),4)+" ":OP$="DB   "+N$:IF AD>A9
 THEN OV=1 ELSE GOSUB 1710
1630 G$=LEFT$(AD$+G$+"          ",13)+"   "+OP$:PRINT G$:IF PR=1 THEN LPRINT G$
1640 LI=LI+1:IF LI=63 THEN LI=1:PRINT:PRINT:IF PR=1 THEN LPRINT CHR$(12);
1690 IF OV=0 THEN 1620 ELSE RETURN
1700 REM DATA 3ﾊﾞｲﾄ ﾖﾐﾄﾘ
1710 BI=1
1720 GOSUB 4610:OP$=OP$+","+N$:BI=BI+1:IF AD>A9 THEN OV=1:BI=4
1730 IF BI<4 THEN 1720 ELSE RETURN
2000 REM ﾒｲﾚｲ ｶｲﾄﾞｸ(AD<adress:ｷﾞｬｸ ｱｾﾝﾌﾞﾗｰ>G$,next adress>AD,end>OV)
2010 GOSUB 3310:AD$=RIGHT$("000"+HEX$(AD),4)+" "+MC$:AD=AD+1:G$="":OP$="":GOSUB
3510:G$=LEFT$(AD$+G$+"          ",13)+"   "+OP$:IF AD>A9 THEN OV=1
2090 RETURN
2100 REM G1=1 ｶｲﾄﾞｸ
2110 ON G3 GOSUB 2510,2610,2710,2810,2910,3010,3110,3210:RETURN
2200 REM G1=2 ｶｲﾄﾞｸ
```

《第 9 図》逆アセンブラ・プログラムリスト

```
2210 IF MC$="76" THEN OP$="HALT" ELSE OP$=A$(8)+C$(G2-1)+","+C$(G3-1)
2290 RETURN
2300 REM G1=3 ｶｲﾄﾞｸ
2310 OP$=A$(G2+22)+C$(G3-1):RETURN
2400 REM G1=4 ｶｲﾄﾞｸ
2410 ON G3 GOSUB 3610,3710,3810,3910,4010,4110,4210,4310:RETURN
2500 REM G1=1 & G3=1
2510 OP$=A$(G2-1):IF G2>2 THEN GOSUB 4510:OP$=OP$+E$
2590 RETURN
2600 REM G1=1 & G3=2
2610 IF G2 AND 1 THEN 2650:REM G2=1,3,5,7—2650
2620 OP$="ADD  HL,"+B$(G2/2-1):GOTO 2690
2650 GOSUB 3410:OP$=A$(8)+B$(G2/2)+","+NN$
2690 RETURN
2700 REM G1=1 & G3=3
2710 OP$=A$(8)
2720 IF G2<5 THEN OP$=OP$+A$(G2+18):GOTO 2790
2730 GOSUB 3410:ON G2-4 GOTO 2740,2750,2760,2770
2740 OP$=OP$+"("+NN$+"),HL":GOTO 2790
2750 OP$=OP$+"HL,("+NN$+")":GOTO 2790
2760 OP$=OP$+"("+NN$+"),A":GOTO 2790
2770 OP$=OP$+"A,("+NN$+")"
2790 RETURN
2800 REM G1=1 & G3=4
2810 IF G2 AND 1 THEN OP$=A$(9)+B$(G2/2) ELSE OP$=A$(10)+B$(G2/2-1)
2890 RETURN
2900 REM G1=1 & G3=5
2910 OP$=A$(9)+C$(G2-1):RETURN
3000 REM G1=1 & G3=6
3010 OP$=A$(10)+C$(G2-1):RETURN
3100 REM G1=1 & G3=7
3110 GOSUB 4610:OP$=A$(8)+C$(G2-1)+","+N$:RETURN
3200 REM G1=1 & G3=8
3210 OP$=A$(G2+10):RETURN
3300 REM ﾏｼﾝ ｺｰﾄﾞ ﾖﾐﾄﾘ(AD<aress,ﾏｼﾝ ｺｰﾄﾞ>MC,MC$)
3310 MC=PEEK(AD):MC$=RIGHT$("0"+HEX$(MC),2)
3390 RETURN
3400 REM nn ｦ ﾓﾄﾒﾙ(AD<adress,nn>NN$,G$=G$+nn,AD=AD+2)
3410 GOSUB 3310:AD=AD+1:G$=G$+MC$:I=MC
3420 GOSUB 3310:AD=AD+1:G$=G$+MC$:NN$=RIGHT$("000"+HEX$(MC*256+I),4)+"H":IF LEFT
$(NN$,1)>"9" THEN NN$="0"+NN$
3490 RETURN
3500 REM ﾒｲﾚｲ ｶｲﾄﾞｸ SUB(AD<adress:ﾏｼﾝ ｺﾎﾄﾞ>G$,ｵﾍﾟﾚｰｼｮﾝ>op$,next adress>AD)
3510 G1=MC¥64+1:G2=((MC¥8) AND 7)+1:G3=(MC MOD 8)+1:ON G1 GOSUB 2110,2210,2310,2
410:RETURN
3600 REM G1=4 & G3=1
3610 OP$="RET  "+D$(G2-1):RETURN
3700 REM G1=4 & G3=2
3710 IF G2 AND 1 THEN IF G2=7 THEN OP$="POP  AF" ELSE OP$="POP  "+B$(G2/2) ELSE
OP$=A$(G2/2+30)
3790 RETURN
3800 REM G1=4 & G3=3
3810 GOSUB 3410:OP$="JP   "+D$(G2-1)+","+NN$:RETURN
3900 REM G1=4 & G3=4
3910 IF G2>4 THEN OP$=A$(G2+30):GOTO 3990
3920 IF G2=1 THEN GOSUB 3410:OP$="JP   "+NN$:GOTO 3990
3930 IF G2=2 THEN GOSUB 4410:GOTO 3990
3940 GOSUB 4610:IF G2=3 THEN OP$="OUT  "+N$+",A" ELSE OP$="IN   A,"+N$
3990 RETURN
4000 REM G1=4 & G3=5
4010 GOSUB 3410:OP$="CALL "+D$(G2-1)+","+NN$:RETURN
4100 REM G1=4 & G3=6
4110 IF G2 AND 1 THEN IF G2=7 THEN OP$="PUSH AF":GOTO 4190 ELSE OP$="PUSH "+B$(G
2/2):GOTO 4190
4120 IF G2=2 THEN GOSUB 3410:OP$="CALL "+NN$:GOTO 4190
```

```
4130 ON G2/2-1 GOSUB 4810,4910,5010
4190 RETURN
4200 REM G1=4 & G3=7
4210 GOSUB 4610:OP$=A$(G2+38)+N$:RETURN
4300 REM G1=4 & G3=6
4310 OP$="RST  "+RIGHT$("0"+HEX$((G2-1)*8),2)+"H":RETURN
4400 REM CB ﾒｲﾚｲ
4410 GOSUB 4610:G1=MC¥64+1:G2=((MC¥8) AND 7):G3=(MC MOD 8):IF G1>1 THEN OP$=A$(G
1+45)+"  "+CHR$(48+G2)+","+C$(G3):GOTO 4490
4420 IF G2=6 THEN OP$="" ELSE OP$=A$(G2+50)+C$(G3)
4490 RETURN
4500 REM e ｦ ﾓﾄﾒﾙ(AD<adress,e>E$,G$=G$+e,AD=AD+1)
4510 GOSUB 3310:AD=AD+1:G$=MC$:IF MC>127 THEN MC=MC-256
4520 E$=RIGHT$("000"+HEX$(AD+MC),4)+"H":IF LEFT$(E$,1)>"9" THEN E$="0"+E$
4590 RETURN
4600 REM n ｦ ﾓﾄﾒﾙ(AD<adress,n>N$,G$=G$+n,AD=AD+1)
4610 GOSUB 3310:AD=AD+1:G$=G$+MC$:N$=MC$+"H":IF LEFT$(N$,1)>"9" THEN N$="0"+N$
4690 RETURN
4700 REM DD & FD ｼｮﾘ (XY$<IX or IY)
4710 GOSUB 4610:IF MC$="DD" OR MC$="ED" OR MC$="FD" THEN 4790
4711 IF MC$="29" THEN OP$="ADD  "+XY$+","+XY$:GOTO 4790
4712 IF MC$="36" THEN GOSUB 4610:OP$="LD   ("+XY$+"+"+N$+"),":GOSUB 4610:OP$=OP$
+N$:GOTO 4790
4713 IF MC$="E9" THEN OP$="JP   ("+XY$+")":GOTO 4790
4714 IF MC$="CB" THEN GOSUB 4610:D$=N$:GOSUB 4410:I=INSTR(OP$,"(HL"):IF I=0 THEN
 OP$="":GOTO 4790 ELSE OP$=LEFT$(OP$,I)+XY$+"+"+D$+")":GOTO 4790
4715 A0=AD:AD=AD-1:GOSUB 3310:AD=AD+1:GOSUB 3510:I=INSTR(OP$,"(HL)"):IF I>0 THEN
 GOSUB 4610:OP$=LEFT$(OP$,I)+XY$+"+"+N$+MID$(OP$,I+3):GOTO 4790
4716 I=INSTR(OP$,"HL"):IF I>0 THEN OP$=LEFT$(OP$,I-1)+XY$+MID$(OP$,I+2) ELSE AD=
A0:G$=LEFT$(G$,2):OP$=""
4790 RETURN
4800 REM DD ﾒｲﾚｲ
4810 XY$="IX":GOSUB 4710:RETURN
4900 REM FD ﾒｲﾚｲ
4910 GOSUB 4610:G1=MC¥64:G2=((MC¥8) AND 7):G3=(MC MOD 8):IF G1=0 OR G1=3 THEN 49
90 ELSE IF G1=1 THEN ON G3+1 GOTO 4920,4922,4924,4926,4928,4930,4932,4934 ELSE 4
950
4920 IF G2=6 THEN 4990 ELSE OP$="IN   "+C$(G2)+",(C)":GOTO 4990
4922 IF G2=6 THEN 4990 ELSE OP$="OUT  (C),"+C$(G2):GOTO 4990
4924 OP$="  HL,"+E$(G2/2):IF G2 AND 1 THEN OP$="ADC"+OP$:GOTO 4990 ELSE OP$="SBC
"+OP$:GOTO 4990
4926 IF G2=4 OR G2=5 THEN 4990 ELSE GOSUB 3410:IF G2 AND 1 THEN OP$="LD   "+B$(G
2/2)+",("+NN$+")":GOTO 4990 ELSE OP$="LD   ("+NN$+"),"+E$(G2/2):GOTO 4990
4928 IF MC$="44" THEN OP$="NEG":GOTO 4990 ELSE 4990
4930 IF G2=0 THEN OP$="RETN":GOTO 4990 ELSE IF G2=1 THEN OP$="RETI":GOTO 4990 EL
SE 4990
4932 IF MC$="46" THEN OP$="IM   0":GOTO 4990 ELSE IF MC$="56" THEN OP$="IM   1":
GOTO 4990 ELSE IF MC$="5E" THEN OP$="IM   2":GOTO 4990 ELSE 4990
4934 IF G2>5 THEN 4990 ELSE OP$=A$(G2+74):GOTO 4990
4950 IF G2>3 OR G3<4 THEN OP$=A$((G2-4)*4+G3+58)
4990 RETURN
5000 REM FD ﾒｲﾚｲ
5010 XY$="IY":GOSUB 4710:RETURN
9000 REM OP ｺｰﾄﾞ
9010 DATA NOP,"EX   AF,AF'","DJNZ ","JR   ","JR   NZ,","JR   Z,","JR   NC,","JR
  C,","LD   ","INC  ","DEC  ",RLCA,RRCA,RLA,RRA,DAA,CPL,SCF,CCF,"(BC),A","A,(BC)
","(DE),A","A,(DE)","ADD  A,","ADC  A,","SUB  ","SBC  A,","AND  ","XOR  ","OR
","CP   "
9030 DATA RET,EXX,JP   (HL),"LD   SP,HL","EX   (SP),HL","EX   DE,HL",DI,EI,"ADD
 A,","ADC  A,","SUB  ","SBC  A,","AND  ","XOR  ","OR   ","CP   ",BIT,RES,SET,"RL
C  ","RRC  ","RL   ","RR   ","SLA  ","SRA  ",,"SRL  "
9045 DATA LDI,CPI,INI,OUTI,LDD,CPD,IND,OUTD,LDIR,CPIR,INIR,OTIR,LDDR,CPDR,INDR,O
TDR,"LD   I,A","LD   R,A","LD   A,I","LD   A,R",RRD,RLD
9100 REM ﾚｼﾞｽﾀ & ﾌﾗｸﾞ
9110 DATA BC,DE,HL,SP,B,C,D,E,H,L,(HL),A,NZ,Z,NC,C,PO,PE,P,M
```

PC-8001/8801N-BASICモード

(注)「スペース・インベーダー」は㈱タイトーの登録商標です。ここで紹介するゲームのプログラムは，同社の「インベーダー」を参考にしてPC-8001／8801N-BASICモード用に独自に作成した技術論文です。

スペース・インベーダーの実際

「マシン語版ゲームへの招待」

はじめに

これから紹介するのは，1978年に登場し現在なお世界中の社会に影響を与えている㈱タイトー発売の

スペース・インベーダー（Ⓒ㈱タイトー）

です。このインベーダーが，日本中にインパクトを与えた当時，マイコン界はまだ**第１世代の後期**でした。ワンボードマイコンから・スイッチ・オンＢＡＳＩＣへの過渡期で，往年の名機

ＴＫ－80
＋ＴＫ－80ＢＳ

の全盛だったのです。これは，

カラーは出ない
キャラクターしか使えない
ハードをいじらないと音は出ない
画面は32×16しかない

というものでしたが，それでもマニアは
「スペース・インベーダー」をやりたい
ということで一歩でもゲーム・センターのそれへ近づけようと努力したものです。

そんなインベーダー・ブームのさなか1979年５月，東京平和島の流通センターで**「マイクロコンピュータショウ'79」**が開催されました。そして

スペース・インベーダー

はこの会場の**隅から隅までを占拠してしまった**のです。それはさながら〝スペース・インベーダー〟の品評会で，いかに自社のマシンが本物の〝インベーダー〟に近づけるかを競っているようでした。

そしてこのとき，インベーダー・ストームの吹き荒れる会場の中で，後年パソコンのベストセラー・マシンとなったＰＣ－8001が姿を現わしたのです。

〝スペース・インベーダー〟への道

それは，当時のマニアに驚愕のまざなしで見られました。この低価格でカラー・グラフィックが使える。これで〝スペース・インベーダー〟ができると。

それでも〝インベーダー〟への道は険しく遠かったのです。音は出るが，ＢＥＥＰ音しか出ない。そして肝心のマシン語に関するノウハウが企業秘密で非公開という厚い壁があったからです。しかしマニア達はその**障壁**を一歩一歩乗り越えていきました。はじめはノロイＢＡＳＩＣ版だけだったものが，だんだんとマシン語版のインベーダーも発表されるようになってきました。今ではマニアにとって，マシン語の壁は壁ではなくなりました。音楽すら日常茶飯時のごとく当り前になってきましたきています。

ここで紹介するインベーダーは，それらのマニアの夢をすべて集大成したものです。

① **オール・マシン語**です。当然スピードは，速すぎるくらいに速いです。
② **電子音・音楽**もふんだんに取り入れました。
●インベーダーの行進音

●ビーム発射音
●インベーダー爆破音
●UFO移動音
●ビーム砲がやられたときの音楽
●最高点が出たときの音楽

以上，すべて取り入れてあります。

③ 細かい点も改良してあります。たとえば，PCが出始めの頃発表されたものでは，スペース・キーを押し放しにすると，ビームが連続発射されました。これではゲームの面白さが半減してしまいます。

ところで以下に御紹介します〝スペース・インベーダー〞，実は**ある目的**をもって製作しました。その点については後述しますが，まずは進んでみてください。

入力から実行まで

プログラムを走らせるのは，簡単です。リストのマシン語をモニタで入力し（最近は，マシン語のゲームが沢山発表されていますから，入力の仕方はもう慣れましたね。わからない方は，前ブロックの各章，また〝PC－8001マシン語活用マニュアル〞〔マイコン'82年1月号別冊付録〕を御覧になってください），

G D000↘

で走らせてください。プログラムのセーブは，

WD000, E3A9↘

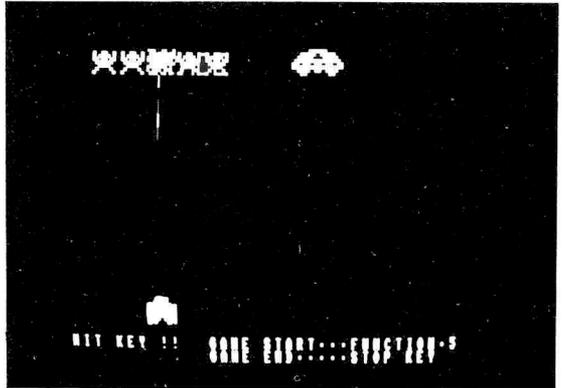
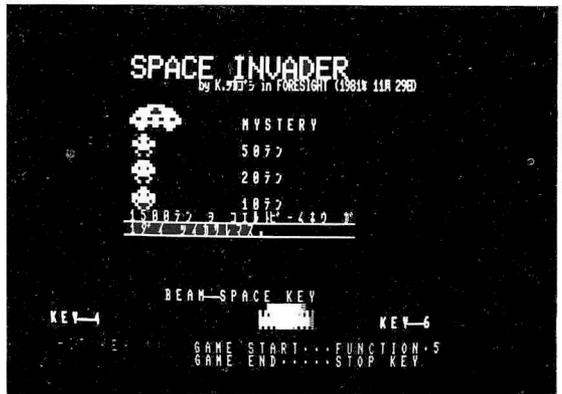
で出来ます。またロードは，

L↘　　です。

デモンストレーション

今回のデモンストレーションの構成は，次の通り。

① 一匹のインベーダーが画面上を動きまわります。
② やがて5匹のインベーダーをつれて行進を始め，画面上部左側に整列します。
③ するとUFOが左側から現われ，インベーダーの右側にならびます。
④ ビーム砲が5匹のインベーダーを撃ちます。すると，〝SPACE〞の文字に変わります。
⑤ ビーム砲がUFOを撃ちます。すると〝INVADER〞の文字に変わります。
⑥ 少しずつ得点類，キー・ファンクション等のゲームの説明が現われます（**写真1，写真2**）。

デモ実行中は，，以上の①－⑥を繰り返す形で進行し

《写真1》 デモ画面

《写真2》 ゲーム説明

ます。デモ中のキー・ファンクションは，次のようになっています。

GAMEを開始する——f･5キー
GAMEをやめる———STOPキー

これは画面上に表示されているように，いつでも受けつけてくれます。なおSTOPキーでは，モニタのコマンド・レベルに戻ります。

遊び方

f・5キーを押すと，**軽快なファンファーレ**とともに

GO!

の文字が現われ(**写真3**)，ゲーム・スタートとなります？遊び方は有名ですから御存知ですね。

各登場人物の動きは，次の通りです。

① **スペース・インベーダー**

隊列を組みながら行進を続け，地球侵略をねらっています。

《写真3》 GO表示

《第1図》 スペース・インベーダーのキーファンクション

ＵＦＯ	50 ～ 300	
インベーダー	赤	50
	黄	20
	緑	10

《第2図》 得 点

《第3図》 画面構成

《写真4》 ゲームスタート直後

② ＵＦＯ

ときどき上方左右に現われます。攻撃はしてきませんが，敵です。

③ **ビーム砲**

あなたが操れる唯一のものです。左右に動かしながらビームを発射し，インベーダーやＵＦＯを攻撃します（キー・ファンクションは**第1図**）。

④ **障 壁**

中立です。インベーダーの放つミサイルをよけるのに利用してください。ミサイル，ビーム，またインベーダーの侵略により壊れていきます。

ゲームの目的は，**インベーダーの侵略を防ぐ**ことにあります。それには，ビーム砲により

すべてのインベーダーを撃破

する必要があります。またときどき現われるＵＦＯは高得点につながりますから余裕があったら狙ってください。得点は，**第2図**のようになっています。また**第3図**に画面の構成図を入れておきます。

インベーダーをすべて撃破すると，一面の終了となります。すると御存知のように一段下の位置からインベーダーが現われ，ゲーム再開となります。五面消すと，インベーダーは最初の位置に戻ります。

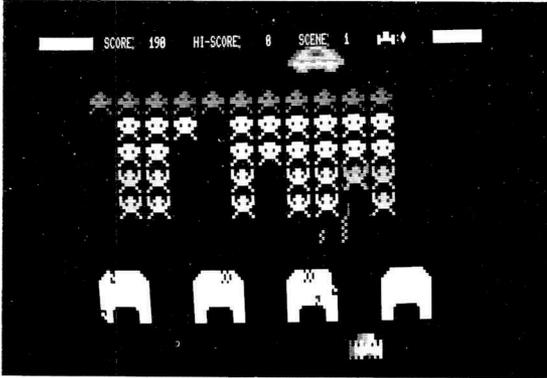
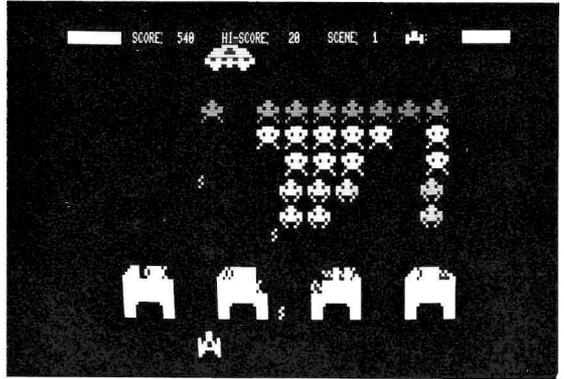
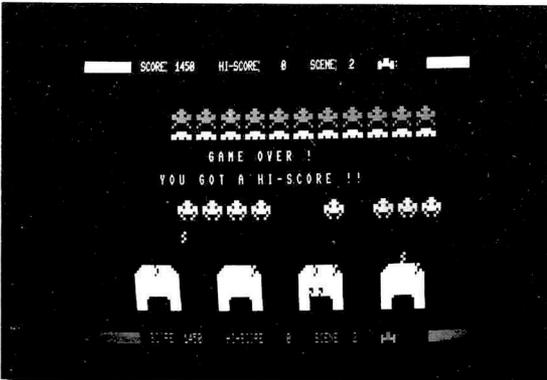
慣れないうちは，一面を消すのはかなり難しいでしょう。というのはインベーダーは数が少なくなるにつれてそのスピードがだんだん速くなるからです。とくに最後の２～３匹になると，メチャクチャに速くなります。私の場合，六面まで行けたのはまだ一回きり！

ＧＡＭＥの終了は，次の二つの場合です。

① **インベーダーに侵略されたとき**

② **ビーム砲がなくなったとき**

ビーム砲の初期値は，３台です。インベーダーの放つミサイルに当ると，悲鳴とともに破壊されます。

逆に得点が1500点を越えると，**ベルの合図**とともにビーム砲が１台追加されます。このベルの音，電子音で構成されていて，なかなか気持ちの良いもの

《写真5》 ＵＦＯ出現

《写真6》 インベーダーと激しい戦い

《写真7》 GEME OVER

です。とくにGAME進行中にこの音が聞こえると，「やった!」と思うと同時に，ホッとします。

ゲームの内容は，以上の通りです（**写真4～6**)。

もう2点ほど注意を申し上げておきます。

① ビームとインベーダーの放つミサイルがぶつかったときは，相撃ちとなり相方が消えます。これを変更するのは簡単ですから，気に入らない人はプログラムを解析して直してください。

② **STOP キーについて**

マシン語で組んだプログラムは，普通ＳＴＯＰキーが効きません（N－ＢＡＳＩＣにおけるＳＴＯＰキーは，ソフトウエアでスキャンしているから，これは当り前ですね)。しかし本プログラムでＳＴＯＰキーを押すと，ＧＡＭＥを一時中断することが出来ます。少し休んだり，作戦をたてるのに利用してください。もう一度ＳＴＯＰキーを押すとＧＡＭＥが再開します。

ＧＡＭＥが終了すると，得点が表示されます。そのとき最高点だと音楽がなり，その旨表示されます（**写真7**)。

その後は，再びデモンストレーションに戻ります。

プログラムについて

プログラムについては，お知らせしておくことがあります。それには，「マイコン」誌〝**GAMINGへの招待**″とからめてお話しした方が良いでしょう。

マイコン各誌はおおむねＢＡＳＩＣ言語を中心にゲーム作りを紹介しています。その中で，**オール・マシン語のゲーム**を取り上げてほしいという意見が多いと聞きました。もっともだと思います。しかし，それは，次の事情により難しいようです。

① 〝ＧＡＭＩＮＧ″への招待は，**すべての人，すべてのマシンを対象**としています。したがって各マシンにおけるハード上の特典を無視するには，ＢＡＳＩＣが便利なのです。

② マシン語の説明は，**大量のページ数**を必要とします。たとえばプリンタでＢＡＳＩＣのリストを取ってもプリンタ用紙はたいして減りません。ところがマシン語のリストを取ると，用紙がまたたくまに減っていくのがわかります。すなわち，マシン語とＢＡＳＩＣでは，**物理的スケールがまったく異なる!**のです。したがってもし〝ＧＡＭＩＮＧへの招待″でオール・マシン語版のゲームを取り上げてしまったら，**ゲーム完結まで何年もかかってしまう**でしょう。

また一方，〝**PC-8001マシン語活用マニュアル**″（前掲および，「PC-8001マシン語入門セミナ開催記」，(付章3として，次章の162頁から紹介しています)の

読者からの貴重な御意見もいただいております。曰く，
「オール・マシン語の解説書は出ないのか？」
「FORESIGHT 版は，手に入らないのか？」
結局私は，次の企画をたてることにしました。
① オール・マシン語版が必要なら，ハッキリ機種はPC-8001(およびPC-8800)に限定してしまおう。
② マシン語の解説をするなら，紙面の枚数を限定することをやめてしまう。
この方針のもとに，次の2点の企画にとりかかりました。
① マシン語の最初の最初から解説したシリーズを作る。
② 一方では，その到達点としての〝オール・マシン語版スペース・インベーダー〟の作り方，解説を誌面の枚数に関係なしに詳細に行う。

①の系統は②にむかって進んでいく。②は①にむかって進んでいく。こうして両者の接点が結ばれたとき，**マシン語の基礎から応用の原点まで**の一つのコースが完成するのでは，またこれにより前記両系統の御意見にお応えできるのでは——こう考えたわけです。

お わ り に

前述のとおり，このオール・マシン語〝スペース・インベーダー〟はひとつの目的をもって制作を進めました。一つ一つのサブルーチンは，できる限りわかりやすく，かつ解析しやすいようにと心掛けました。
将来このプログラムを解析してみようと思われる方は，ひとまずこのプログラムを入力し，ぜひ遊んでみてください。
最後に次頁から待望の〝スペース・インベーダー〟のプログラムを示します。長いプログラムですから入力ミスのないようにガンバッて下さい。念のためにチェックサムのプログラムも下に示しましたので使ってみて下さい。
まずプログラムを入力し，RUNさせると，チェックサムをとるスタートアドレスをきいてきますから，インベーダーの場合ならば，D000と入力して下さい。
次々に，チェックサムが表示されていきます。一時停止はエスケープキーで行う（ESCキー）とよいでしょう。プログラムは次のようになっています。

10：〜30：コメント
40：80文字モード選択
50：スタートアドレス入力
60：十六進数値へ変換
70：総和を求める変数SUMを0にする
80：〜90：一行の総和を求める
110：区間とチェックサムをプリントアウトする
120：次の区間の先頭アドレス
130：次のステップへ

CPU	8080（2MHZ）	
ROM	2708×7（7Kバイト）	
RAM	ワーク・エリア	2112×2（1Kバイト）
	ビデオRAM	2114×16（8Kバイト）〔タテ8ドットを1バイトで表わしている〕

《第4図》 スペース・インベーダー（オリジナル）の構成

〈プログラム1〉PC-8001/8801(N-BASIC)チェックサム・プログラム

```
10 '
20 'Check sum program for PC-8001 (N-BASIC)
30 '
40 WIDTH 80
50 INPUT "Check sum from ";ADDR$
60 STA=VAL("&H"+ADDR$)
70 SUM=0
80 FOR I=STA TO STA+15
90    SUM=SUM+PEEK(I)
100 NEXT I
110 PRINT HEX$(STA);" - ";HEX$(STA+15);" Sum=";RIGHT$("0000"+HEX$(SUM),4)
120 STA=STA+16
130 GOTO 70
```

《第1表》 スペース・インベーダー・プログラムリスト

```
D000: CD 93 DC CD 9F DC CD C3 DC CD FF DC CD 13 D2 3A ; 0B84
D010: AA E3 3D 28 F4 3D 28 EE CD B2 DD 18 E6 00 00 00 ; 0793
D020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; 0000
D030: 00 00 CD E2 D8 CD 6C D2 CD A0 D2 CD C3 D2 CD F6 ; 0AF6
D040: D2 CD 1D D3 CD 72 D3 CD A7 D3 CD D4 D3 06 14 CD ; 0A43
D050: 3D D4 21 13 16 22 AD E3 06 02 CD 58 D4 CD 74 D4 ; 0723
D060: 06 30 CD 3D D4 3E 32 32 AE E3 06 02 CD 58 D4 CD ; 0715
D070: C8 D0 CD ED D0 CD FD D0 CD ED D0 CD 10 D1 CD ED ; 0CAE
D080: D0 CD 2E D1 CD ED D0 CD 67 D1 CD ED D0 CD 7C D1 ; 0BCF
D090: CD ED D0 CD E6 D1 CD ED D0 CD 91 D1 3E 03 32 AB ; 0AE5
D0A0: E3 CD CC D1 21 13 11 22 AD E3 06 03 CD 58 D4 06 ; 074C
D0B0: 3C CD 3D D4 21 13 3E 22 AD E3 06 03 CD 58 D4 21 ; 0661
D0C0: AB E3 35 20 DC C3 32 D0 25 25 2D CD 20 D7 AF 01 ; 076F
D0D0: 02 09 21 01 2E CD 4E D9 21 50 B8 22 CA F3 22 42 ; 05BB
D0E0: F4 01 02 15 11 9F E2 21 01 26 C3 9C DD 0E 0A 06 ; 0540
D0F0: 32 CD 85 D9 CD 8C D2 10 F8 0D 20 F3 C9 3E 68 2E ; 084D
D100: 03 CD 60 D9 21 03 20 CD 0C D9 11 C9 E2 C3 27 D9 ; 077E
D110: 11 F1 E2 21 A9 F5 CD 1D D9 21 05 14 22 CD E3 CD ; 083F
D120: 1F DB 21 06 28 CD 0C D9 11 FB E2 C3 27 D9 11 09 ; 06C6
D130: E3 2E 07 1A A7 28 1B D5 06 02 E5 C5 F5 CD 14 D9 ; 0752
D140: F1 C1 23 77 23 36 28 23 36 E8 E1 2C 10 EC D1 13 ; 06FB
D150: 18 E1 11 AD DF 21 07 14 CD 72 D5 21 08 28 CD 0C ; 0610
D160: D9 11 0D E3 C3 27 D9 11 C5 DF 21 09 14 CD 72 D5 ; 07A4
D170: 21 0A 28 CD 0C D9 11 15 E3 C3 27 D9 11 DD DF 21 ; 06BF
D180: 0B 14 CD 72 D5 21 0C 28 CD 0C D9 11 1D E3 C3 27 ; 0635
D190: D9 3E E8 32 29 FD 21 40 E8 22 32 FD 2E 13 CD 14 ; 0713
D1A0: D9 23 11 55 E3 CD 27 D9 21 15 05 CD 0C D9 11 25 ; 0635
D1B0: E3 CD 27 D9 21 15 41 CD 0C D9 11 2F E3 CD 27 D9 ; 07C9
D1C0: 21 13 1A CD 0C D9 11 39 E3 C3 27 D9 CD 12 D4 21 ; 06C4
D1D0: AC E3 35 CD F3 D3 06 14 CD 73 D9 CD 8C D2 3A AC ; 099B
D1E0: E3 FE 0F 20 E7 C9 06 02 2E 0D C5 E5 CD 14 D9 23 ; 078A
D1F0: 11 5B E3 CD 27 D9 E1 C1 2C 10 EF 21 0D 14 CD 0C ; 0704
D200: D9 11 66 E3 CD 27 D9 21 0E 14 CD 0C D9 11 90 E3 ; 0779
D210: C3 27 D9 3A AA E3 A7 C0 CD F1 0C DC 66 D2 3A B0 ; 09B9
D220: E3 47 3A AB E3 A0 CC DE D4 3A AB E3 E6 0F CC CB ; 0A64
D230: DA 3A B0 E3 47 3A AB E3 A0 CC 3E DE 3A AB E3 E6 ; 09EC
D240: 07 CC 36 DB 3A AB E3 E6 03 CC 7B DB 3A AB E3 E6 ; 0965
D250: 0F CC C8 DE CD B2 D7 CD 19 D8 21 AB E3 34 CD 1D ; 0962
D260: DD CD 85 D9 18 AD CD F1 0C 30 FB C9 21 19 FE 11 ; 08D4
D270: DB E1 CD 27 D9 21 91 FE 13 CD 27 D9 21 CE FD 11 ; 0916
D280: E7 E1 CD 27 D9 21 5E FE 13 C3 27 D9 DB 09 FE FE ; 09C8
D290: CA 66 5C FE DF C0 F1 F1 C9 00 00 00 00 00 00 00 ; 06D4
D2A0: 21 C9 F3 11 59 E2 CD 27 D9 21 41 F4 11 59 E2 CD ; 0865
D2B0: 27 D9 3E 58 2E 03 06 06 C5 E5 CD 60 D9 E1 C1 2C ; 0751
D2C0: 10 F6 C9 11 DD DF DD 21 60 E2 AF 32 B9 E3 DD 7E ; 09B4
D2D0: 00 A7 C8 D5 67 DD 7E 01 6F E5 CD 72 D5 DD 23 DD ; 094C
D2E0: 23 21 B9 E3 7E 2F 77 06 50 CD 73 D9 E1 CD 6B D5 ; 0861
D2F0: D1 CD 8C D2 18 D8 11 DD DF 21 01 2E 7C FE 4C C8 ; 0897
D300: D5 E5 CD 72 D5 06 3C CD 73 D9 E1 E5 CD 6B D5 21 ; 0A1D
D310: B9 E3 7E 2F 77 E1 D1 24 CD 8C D2 18 DF 01 00 01 ; 07BA
D320: 21 01 4B 78 FE 06 C8 E5 CD 49 D3 C5 06 32 CD 73 ; 07BC
D330: D9 21 B9 E3 7E 2F 77 C1 E1 E5 CD 62 D3 E1 25 78 ; 09C1
D340: 81 47 0E 00 CD 8C D2 18 DA C5 C5 E5 11 DD DF CD ; 08FC
D350: 72 D5 E1 C1 7C C6 05 67 10 F0 C1 3A C3 F3 A7 C0 ; 09AF
D360: 0C C9 C5 C5 E5 CD 6B D5 E1 C1 7C C6 05 67 10 F3 ; 09A4
D370: C1 C9 21 01 31 7C FE 0A C8 E5 06 05 CD 62 D3 E1 ; 07FC
D380: 25 E5 CD 90 D3 06 32 CD 73 D9 E1 CD 8C D2 18 E5 ; 0994
D390: C5 E5 11 DD DF CD 72 D5 E1 C1 7C C6 05 67 10 F0 ; 09DB
D3A0: 21 B9 E3 7E 2F 77 C9 21 23 98 22 CC F3 26 38 22 ; 06E7
D3B0: 44 F4 21 01 47 22 CD E3 CD 2C DB 21 CE E3 7E 3D ; 07D4
D3C0: 77 CD 1F DB 06 40 CD 73 D9 CD 8C D2 3A CE E3 FE ; 09B1
D3D0: 2E 20 E5 C9 3E 4C 32 AC E3 CD 12 D4 21 AC E3 35 ; 07DF
D3E0: CD F3 D3 06 0F CD 73 D9 CD 8C D2 3A AC E3 FE 05 ; 09B8
D3F0: 20 E7 C9 2E 14 CD 14 D9 3A AC E3 47 C5 11 02 E0 ; 0794
D400: 23 23 CD 1F D4 C1 60 2E 14 CD 0C D9 11 0C E0 C3 ; 06DB
D410: 1D D9 01 02 04 3A AC E3 67 2E 14 AF C3 4E D9 C5 ; 06CD
D420: E5 CD 2B D4 E1 01 78 00 09 C1 13 70 23 1A 77 04 ; 0610
D430: 13 23 1A A7 C8 70 23 77 04 13 23 18 F5 C5 CD 12 ; 05B4
D440: D4 21 AC E3 34 CD F3 D3 06 0F CD 73 D9 C1 CD 8C ; 0993
D450: D2 3A AC E3 B8 20 E6 C9 2A AD E3 C5 E5 CD 0C D9 ; 0A38
D460: 36 0F 06 0F CD 73 D9 36 00 E1 C1 2D CD 8C D2 7D ; 0720
D470: B8 20 E8 C9 25 25 2D CD 20 D7 AF 01 02 19 21 01 ; 05B1
D480: 0A CD 4E D9 3E B8 32 CB F3 32 43 F4 01 02 0F 11 ; 0670
D490: 81 E2 21 01 14 C3 9C DD 3A AC E3 A7 C8 CD D1 D4 ; 097F
D4A0: 21 AC E3 35 18 0D 3A AC E3 FE 4C C8 CD D1 D4 21 ; 0878
D4B0: AC E3 34 2E 17 CD 14 D9 3A AC E3 47 C5 11 02 E0 ; 078A
D4C0: CD 2F D9 C1 60 2E 17 CD 0C D9 11 0C E0 C3 1D D9 ; 07A3
D4D0: 00 01 02 04 3A AC E3 67 2E 17 AF C3 4E D9 AF 32 ; 05F6
D4E0: C4 E3 21 FA D4 E5 3A BA E3 A7 CA 07 D5 FE 01 CA ; 0A68
D4F0: F3 D5 FE 02 CA 52 D6 C3 9A D6 3A C4 E3 A7 C0 AF ; 0AE4
D500: 2A C2 E3 23 77 18 D7 2A C2 E3 7E FE FF CA 95 D5 ; 09D6
D510: A7 20 05 CD 32 D5 18 EF 06 0B C5 CD 4A D5 30 08 ; 06A1
D520: 25 ED 5B BF E3 CD 72 D5 21 BE E3 7E C6 05 77 C1 ; 0966
D530: 10 E8 21 BD E3 35 35 3A BC E3 32 BE E3 21 C1 E3 ; 0894
D540: 35 7E CD 87 D5 21 C2 E3 35 C9 2A BD E3 CD 5D D5 ; 0969
D550: D0 E5 CD 6B D5 E1 3E FF 32 C4 E3 37 C9 E5 24 24 ; 09E6
D560: CD 0C D9 7E E1 FE EE 37 C8 3F C9 01 02 05 AF C3 ; 087E
D570: 4E D9 D5 CD 0C D9 D1 3A B9 E3 A7 CA 1D D9 EB 01 ; 09A8
D580: 0C 00 09 EB C3 1D D9 E6 01 C8 2A BF E3 01 E8 FF ; 081C
D590: 09 22 BF E3 C9 21 BC E3 35 21 C4 E3 36 FF CD AF ; 0904
D5A0: D5 CD D0 D5 26 00 CD D6 D5 C8 21 BA E3 34 C9 21 ; 0989
D5B0: B9 E3 7E 2F 77 C3 B8 D5 2A BB E3 22 BD E3 21 DD ; 0998
D5C0: DF 22 BF E3 3E 05 32 C1 E3 21 B8 E3 22 C2 E3 C9 ; 0908
D5D0: 01 03 06 C3 15 DA AF 32 C5 E3 3A BB E3 6F 06 05 ; 0697
D5E0: CD 5D D5 30 05 3E FF 32 C5 E3 2D 2D 10 F2 3A C5 ; 07A6
D5F0: E3 A7 C9 2A C2 E3 7E FE FF 28 42 A7 20 05 CD 32 ; 08D2
D600: D5 18 F0 06 0B C5 CD 4A D5 30 08 2C ED 5B BF E3 ; 07ED
D610: CD 72 D5 21 BE E3 7E C6 05 77 C1 10 E8 CD 32 D5 ; 0923
D620: CD D0 D5 2A BD E3 2C 2C 2C 2C 3A C1 E3 A7 28 08 ; 07A1
D630: FE 03 D0 3E D8 C3 60 D9 3E 58 C3 60 D9 21 BB E3 ; 0934
D640: 34 3E 32 23 77 21 C4 E3 36 FF CD AF D5 21 BA E3 ; 084A
D650: 34 C9 2A C2 E3 7E FE FF 28 26 A7 20 05 CD 32 D5 ; 0835
D660: 18 F0 06 0B C5 CD 4A D5 30 08 24 ED 5B BF E3 CD ; 07DD
D670: 72 D5 21 BE E3 7E D6 05 77 77 C1 10 E7 C3 32 D5 ; 08D2
D680: 21 BC E3 34 21 C4 E3 36 FF CD AF D5 CD D0 D5 26 ; 09DA
```

```
D690: 4B CD D6 D5 C8 21 BA E3 34 C9 2A C2 E3 7E FE FF ; 0A90
D6A0: 28 2B A7 20 05 CD 32 D5 18 F0 06 0B C5 CD 4A D5 ; 06BD
D6B0: 30 08 2C ED 5B BF E3 CD 72 D5 21 BE E3 7E D6 05 ; 087D
D6C0: 77 C1 10 E8 CD 32 D5 CD D0 D5 C3 23 D6 21 BB E3 ; 09F1
D6D0: 34 3E 19 23 77 21 C4 E3 36 FF CD AF D5 AF 32 BA ; 080E
D6E0: E3 C9 CD 03 D7 E5 CD 20 D7 E1 E5 CD 6B D5 E1 CD ; 0B7D
D6F0: 4A D7 21 00 00 22 AD E3 21 B1 E3 35 C0 3E 02 32 ; 0610
D700: AA E3 C9 2A AD E3 E5 CD 0C D9 7E E6 0F E1 28 03 ; 0926
D710: 25 18 F3 E5 24 24 CD 0C D9 7E E1 FE EE C8 2D C9 ; 0918
D720: CD 0C D9 11 16 E0 E5 CD 1D D9 CD 3E D7 06 10 CD ; 0826
D730: 73 D9 E1 11 22 E0 CD 1D D9 06 20 C3 73 D9 CD 4E ; 0853
D740: 0D 06 0A CD 73 D9 AF C3 4B 0D 3A BB E3 D6 03 BD ; 076E
D750: 38 07 D6 04 BD 38 07 18 0A 01 0A 00 18 08 01 14 ; 0277
D760: 00 18 03 01 32 00 2A C6 E3 09 22 C6 E3 C3 7C D8 ; 060C
D770: 3E 37 32 B1 E3 21 B2 E3 34 01 01 05 21 B8 E3 71 ; 0659
D780: 2B 10 FC AF 32 B9 E3 32 BA E3 26 0C 3A B2 E3 6F ; 07F3
D790: 22 BB E3 CD A1 D7 CD B2 D7 CD B8 D5 CD C7 D7 18 ; 0B38
D7A0: 42 CD 95 D9 E6 01 C8 3E 02 32 BA E3 3E 3E 32 BC ; 07A5
D7B0: E3 C9 3A B1 E3 06 FF 17 38 04 CB 18 18 F9 78 3F ; 077D
D7C0: 1F E6 0F 32 B0 E3 C9 11 A6 DF 2E 16 1A A7 C8 47 ; 074C
D7D0: 13 1A D5 F5 C5 E5 CD 60 D9 E1 C1 F1 2D 10 F4 D1 ; 0A3C
D7E0: 13 18 E9 06 05 11 DD DF 26 0C 3A BD E3 6F C5 E5 ; 0711
D7F0: CD 07 D8 E1 C1 78 E6 01 20 08 7B D6 18 5F 7A DE ; 07F5
D800: 00 57 2D 2D 10 E8 C9 06 0B C5 D5 E5 CD 72 D5 E1 ; 07F7
D810: D1 C1 7C C6 05 67 10 F1 C9 AF 32 C5 E3 21 16 02 ; 07CC
D820: CD 0C D9 06 4C 7E FE EE 20 05 3E FF 32 C5 E3 23 ; 07CD
D830: 10 F3 3A C5 E3 A7 C8 3E 03 32 AA E3 2E 16 06 03 ; 06A1
D840: C5 E5 3E 58 CD 60 D9 E1 E5 3E 3C CD 6E D8 E1 C1 ; 0A3B
D850: 2C 10 ED 06 05 C5 3E 21 D3 51 01 50 10 CD 15 DA ; 0599
D860: 3E 20 D3 51 01 50 60 CD 0D DA C1 10 E8 C9 4F CD ; 0785
D870: FD D8 EB 06 50 79 B6 77 23 10 FA C9 11 00 12 2A ; 06FF
D880: C6 E3 C3 B6 D9 11 00 25 2A C8 E3 C3 B6 D9 11 00 ; 0869
D890: 35 2A CA E3 C3 A9 D9 01 20 00 11 50 F3 21 94 E0 ; 075B
D8A0: ED B0 01 50 00 11 00 F3 21 B4 E0 ED B0 CD 7C D8 ; 0865
D8B0: 18 D3 21 12 0A CD C7 D8 21 12 1B CD C7 D8 21 12 ; 0681
D8C0: 2C CD C7 D8 21 12 3D 0E 04 11 05 E1 06 09 E5 D5 ; 05DA
D8D0: C5 CD 0C D9 C1 D1 1A 77 23 13 10 FA E1 2C 0D 20 ; 0714
D8E0: EB C9 01 19 50 CD 3A 09 21 18 01 22 5D EA 01 FF ; 05D1
D8F0: 00 CD F7 08 21 00 F8 22 5A EA C3 5A 04 E5 7D 07 ; 06D5
D900: 5F 16 00 21 75 06 19 5E 23 56 E1 C9 CD FD D8 6C ; 06B9
D910: 26 00 19 C9 CD FD D8 EB 01 50 00 09 C9 E5 CD 27 ; 0791
D920: D9 E1 01 78 00 09 13 1A A7 C8 77 13 23 18 F8 C5 ; 065A
D930: E5 CD 3B D9 E1 01 78 00 09 C1 13 AF 77 23 1A 77 ; 06D7
D940: 04 13 23 1A A7 C8 70 23 77 04 13 23 18 F5 E5 C5 ; 05BE
D950: F5 CD 0C D9 F1 77 23 10 FC C1 E1 2C 0D 20 EF C9 ; 08F1
D960: F5 CD 14 D9 AF 77 F1 23 77 01 50 13 23 71 23 77 ; 06F2
D970: 10 FA C9 CD 85 D9 10 FB C9 11 FF FF 1D C2 7C D9 ; 0A15
D980: 15 C2 7C D9 C9 1E FF 1D 20 FD C9 CD 95 D9 90 28 ; 0908
D990: 02 30 FB 80 C9 E5 2A CB E3 7C 85 6F 7C C6 03 67 ; 084F
D9A0: 7D CE 00 6F 22 CB E3 E1 C9 C5 E5 EB CD 0C D9 EB ; 0A66
D9B0: E1 D5 26 00 18 15 C5 E5 EB CD 0C D9 EB E1 D5 01 ; 08F2
D9C0: 10 27 CD EF D9 01 E8 03 CD EF D9 01 64 00 CD EF ; 086E
D9D0: D9 0E 0A CD EF D9 7D C6 30 12 E1 7E FE 30 20 05 ; 07BD
D9E0: 36 20 23 18 F6 1A FE 20 20 03 3E 30 12 C1 C9 C5 ; 05B1
D9F0: AF 3E 30 ED 42 38 03 3C 18 F9 C1 09 12 13 C9 F5 ; 0681
DA00: 3E FF CD 1E DA 3D 20 FA 0D 20 F5 F1 C9 CD 1E DA ; 08FA
DA10: 04 0D 20 F9 C9 CD 1E DA 05 05 0D 20 F8 C9 D5 50 ; 06D5
DA20: 3A 67 EA CB EF D3 40 15 20 FB 50 3A 67 EA CB AF ; 08DD
DA30: D3 40 15 20 FB D1 C9 7E A7 C8 47 23 4E 07 30 05 ; 06BE
DA40: CD 58 DA 18 03 CD 8D DA 3A 67 EA CB AF D3 40 06 ; 086C
DA50: 10 CD 85 D9 10 FB 18 DF E5 D5 CD 64 DA D1 0D 20 ; 0900
DA60: F8 E1 23 C9 EB 50 3A 67 EA CB AF D3 40 2B 7C A7 ; 0966
DA70: 28 15 15 20 F8 50 3A 67 EA CB AF D3 40 2B 7C A7 ; 0720
DA80: 28 05 15 20 F8 18 DE 3A 67 EA CB AF C9 E5 D5 CD ; 08A5
DA90: 99 DA D1 0D 20 F8 E1 23 C9 EB 50 3A 67 EA CB EF ; 09B6
DAA0: D3 40 2B 7C A7 28 15 15 20 F8 50 3A 67 EA CB AF ; 0720
DAB0: D3 40 2B 7C A7 28 05 15 20 F8 18 DE C9 3E 98 2E ; 067E
DAC0: 01 CD 60 D9 3E 38 2E 02 C3 60 D9 2A CD E3 7C B5 ; 07B4
DAD0: 20 1E CD 95 D9 E6 7F C0 CD 95 D9 E6 01 20 03 67 ; 084A
DAE0: 18 04 3E FF 26 47 32 CF E3 2E 01 22 CD E3 18 2F ; 05F2
DAF0: CD 2C DB 21 CE E3 3A CF E3 A7 20 0F 7E FE 47 20 ; 084B
DB00: 07 21 00 00 22 CD E3 C9 34 18 0C 7E A7 20 07 21 ; 0488
DB10: 00 00 22 CD E3 C9 35 01 06 20 CD 0D DA 18 00 2A ; 04ED
DB20: CD E3 CD 0C D9 11 29 E1 1A C3 1D D9 01 02 09 AF ; 070B
DB30: 2A CD E3 C3 4E D9 DB 00 FE EF CC 98 D4 FE BF CC ; 0B4D
DB40: A6 D4 CD 49 DB DC 60 DB C9 DB 09 FE BF 37 3F 28 ; 098A
DB50: 05 AF 32 AF E3 C9 21 AF E3 7E A7 C0 2F 77 37 C9 ; 087F
DB60: 2A AD E3 7C B5 C0 CD 75 DB 3A AC E3 3C 3C 67 2E ; 089E
DB70: 17 22 AD E3 C9 01 1A 30 C3 0D DA 2A AD E3 7C B5 ; 0772
DB80: C8 E5 CD A7 DB E1 2D CD B1 DB 22 AD E3 CD 0C D9 ; 0AC7
DB90: 7E A7 20 03 36 0F C9 E5 F5 CD BC DB F1 E1 CD 34 ; 0967
DBA0: DC CD 88 DC C3 E2 D6 CD 0C D9 7E FE 0F C0 AF 77 ; 0AAB
DBB0: C9 7D A7 C0 21 00 00 22 AD E3 E1 C9 3A AD E3 FE ; 08F2
DBC0: 03 D0 CD EB DB 3E E8 2E 02 CD 60 D9 CD 00 DC CD ; 0938
DBD0: 85 D9 CD 79 D9 CD BD DA CD 2C DB CD 7C D8 21 00 ; 09F7
DBE0: 00 22 CD E3 22 AD E3 E1 E1 E1 C9 2A AD E3 2D 25 ; 08FC
DBF0: 25 E5 CD 20 D7 E1 AF 01 02 05 CD 4E D9 C3 2C DB ; 0824
DC00: CD 95 D9 E6 07 28 17 FE 03 38 0E FE 05 38 05 21 ; 060F
DC10: 32 00 18 0D 21 64 00 18 08 21 96 00 18 03 21 2C ; 021B
DC20: 01 E5 ED 5B CD E3 1C CD B6 D9 E1 ED 4B C6 E3 09 ; 0A21
DC30: 22 C6 E3 C9 CD 3F DC 21 00 00 22 AD E3 E1 C9 FE ; 08F7
DC40: 8C 28 26 FE C8 28 22 FE 42 28 1E FE 77 28 1D FE ; 0728
DC50: 7E 28 19 FE 96 28 15 FE F7 28 11 FE EF 28 11 FE ; 07E3
DC60: FE 28 0D FE FF 28 09 E1 C9 AF 18 06 3E 42 18 02 ; 0672
DC70: 3E 96 77 47 C9 3E 78 2E 12 06 05 F5 C5 E5 CD 60 ; 0728
DC80: D9 E1 C1 F1 2C 10 F4 C9 FE 5A C0 21 00 00 22 AD ; 086D
DC90: E3 E1 C9 3E FF 32 B3 E3 21 00 00 22 C8 E3 C9 CD ; 0916
DCA0: 32 D0 CD 64 DD CD E2 D8 3E 0A 32 B2 E3 21 00 00 ; 07C7
DCB0: 22 C6 E3 22 CA E3 CD 97 D8 3E 03 32 D0 E3 AF 32 ; 08DD
DCC0: E6 E3 C9 CD E2 D8 CD 97 D8 21 B2 E3 7E FE 0F 38 ; 0ACE
DCD0: 02 36 0A CD 70 D7 CD 11 DD 3E FF 32 B9 E3 21 CA ; 0807
DCE0: E3 34 CD 8E D8 CD 75 DC CD B2 D8 CD BD DA AF 32 ; 0B04
DCF0: D1 E3 21 00 00 22 CD E3 22 AD E3 22 D2 E3 C9 3E ; 0837
DD00: 28 32 AC E3 AF 32 AA E3 32 AF E3 CD 4A DD C3 B3 ; 0985
DD10: D4 3A B2 E3 D6 03 6F 11 A8 DF C3 CC D7 3A E6 E3 ; 09EC
DD20: A7 C0 2A C6 E3 11 DC 05 CD D3 5E D8 21 D0 E3 34 ; 090A
```

```
DD30: 21 E6 E3 7E 2F 77 06 32 C5 01 20 15 CD FF D9 06 ; 06EC
DD40: 08 CD 73 D9 C1 10 F1 C3 4A DD 21 00 41 CD 0C D9 ; 07E1
DD50: E5 06 03 36 00 23 10 FB E1 06 EA 3A D0 E3 3D C8 ; 0715
DD60: 70 23 18 FA CD E2 D8 AF D3 51 11 FF 22 21 89 E1 ; 08BC
DD70: CD 37 DA 06 28 C5 01 1A 2D CD FF D9 CD 85 D9 C1 ; 08AA
DD80: 10 F3 01 19 50 CD 3A 09 01 02 0C 11 9E E1 21 14 ; 0451
DD90: 14 CD 9C DD 06 04 CD 79 D9 10 FB C9 C5 E5 C5 D5 ; 099B
DDA0: CD 0C D9 D1 C1 1A 77 13 23 10 FA E1 C1 2C 0D 20 ; 0710
DDB0: EB C9 CD 13 DE CD CD DD 2E 08 3E E8 CD 60 D9 21 ; 096C
DDC0: 08 1A CD 0C D9 11 67 E1 CD E6 DD 18 2D 21 07 00 ; 062A
DDD0: CD 0C D9 06 05 E5 0E 78 36 00 23 0D 20 FA E1 11 ; 059A
DDE0: 78 00 19 10 F0 C9 1A A7 C8 77 13 23 23 FE 20 28 ; 05F9
DDF0: F5 D5 E5 CD 79 D9 E1 D1 18 EC 3A AB E3 A7 C8 2E ; 0AE9
DE00: 0A 3E A8 CD 60 D9 21 0A 10 CD 0C D9 11 73 E1 CD ; 0715
DE10: E6 DD C9 AF 32 AB E3 ED 5B C6 E3 2A C8 E3 CD D3 ; 0B61
DE20: 5E D0 ED 53 C8 E3 AF D3 51 11 FF 35 21 2D E0 CD ; 092C
DE30: 37 DA 01 19 50 CD 3A 09 3E FF 32 AB E3 C9 3A D1 ; 075C
DE40: E3 FE 03 D0 3A B1 E3 FE 0C 38 06 3A AB E3 E6 1F ; 0897
DE50: C0 CD 95 D9 E6 01 28 4F 3A AC E3 3C 67 3A BB E3 ; 089D
DE60: 6F 06 05 C5 E5 CD 6F DE E1 C1 2D 2D 10 F5 C9 E5 ; 08ED
DE70: CD 0C D9 7E 21 3D E1 01 16 00 ED B1 E1 C0 2C 2C ; 071D
DE80: DD 21 D2 E3 DD 7E 00 A7 28 06 DD 23 DD 23 18 F4 ; 07EF
DE90: DD 75 00 DD 74 01 DD 36 02 00 00 00 00 00 00 21 ; 03DA
DEA0: D1 E3 34 E1 E1 E1 C9 2A BB E3 24 24 3A BA E3 E6 ; 0A21
DEB0: 02 28 04 7C D6 37 67 06 0B CD 8B D9 A7 28 A2 F5 ; 06C6
DEC0: 7C C6 05 67 F1 3D 18 F4 DD 21 D2 E3 DD 7E 00 A7 ; 089D
DED0: C8 CD DA DE DD 23 DD 23 18 F2 DD 6E 00 DD 66 01 ; 08E6
DEE0: CD 0C D9 7E FE 5A 20 02 36 00 01 78 00 09 7E A7 ; 0587
DEF0: 20 0A 36 5A DD 7E 00 3C DD 77 00 C9 CD 3A DF DD ; 0731
DF00: 7E 00 FE 16 38 07 FE 18 30 03 CD 40 DF DD E5 FD ; 07C5
DF10: E1 FD 7E 02 A7 28 12 FD 7E 02 FD 77 00 FD 7E 03 ; 07AE
DF20: FD 77 01 FD 23 FD 23 18 E8 FD 36 00 00 FD 36 01 ; 071C
DF30: 00 21 D1 E3 35 DD 2B DD 2B C9 CD 3F DC E1 18 CD ; 0891
DF40: DD 6E 00 2C DD 66 01 CD 0C D9 7E 01 09 00 21 0C ; 0522
DF50: E0 ED B1 C0 CD 69 DF 3E 01 32 AA E3 21 D0 E3 35 ; 095A
DF60: C0 3E 03 32 AA E3 C3 53 D8 06 07 C5 11 53 E1 CD ; 0792
DF70: 87 DF 11 5D E1 CD 87 DF C1 10 F0 CD D1 D4 11 FF ; 0A2B
DF80: 17 21 70 E0 C3 37 DA 2E 17 3A AC E3 67 D5 CD 0C ; 077F
DF90: D9 D1 CD 1D D9 CD 95 D9 F6 07 47 0E 14 CD 0D DA ; 08C2
DFA0: 06 0A CD 73 D9 C9 0E 98 04 D8 02 58 00 80 48 EE ; 0684
DFB0: 84 08 00 90 35 33 53 09 00 80 48 EE 84 08 00 10 ; 0432
DFC0: B5 33 5B 01 00 C0 6C EE C6 0C 00 80 35 33 53 08 ; 0573
DFD0: 00 80 6C EE C6 08 00 10 B5 33 5B 01 00 E0 48 EE ; 0612
DFE0: 84 0E 00 80 35 33 53 08 00 80 48 EE 84 08 00 30 ; 0447
DFF0: B5 33 5B 03 00 01 03 08 09 10 30 33 35 53 5B 80 ; 0331
E000: 90 B5 B8 F8 F8 B8 00 B8 38 38 B8 00 0C 78 87 C0 ; 0850
E010: 00 7F 23 32 F7 00 A1 0C B8 4E 74 00 84 3F 11 43 ; 0509
E020: 84 00 10 87 AE 03 00 22 58 B6 13 1F 00 22 04 28 ; 037C
E030: 03 26 01 22 04 33 04 2D 01 28 01 26 01 22 01 26 ; 014E
E040: 02 28 02 2D 01 44 01 44 01 3D 01 44 01 4C 01 51 ; 0205
E050: 01 5B 01 28 01 26 01 22 01 1E 01 22 02 22 02 19 ; 0150
E060: 04 22 04 26 02 28 01 2D 01 2D 03 33 01 33 06 00 ; 0146
```

```
E070: 1E 04 1E 03 1E 01 1E 04 19 03 1B 01 1B 03 1E 01 ; 00F9
E080: 1E 03 20 01 1E 08 00 33 01 2D 01 28 01 22 02 28 ; 013F
E090: 01 22 04 00 00 28 0A A8 11 58 12 E8 1C A8 24 58 ; 03A4
E0A0: 25 E8 2F A8 34 58 35 E8 3D B8 3E 38 3F B8 40 E8 ; 0717
E0B0: 41 C8 45 28 87 87 87 87 87 87 87 87 87 87 20 20 ; 06FC
E0C0: 53 43 4F 52 45 09 39 39 39 39 39 20 20 20 20 20 ; 0342
E0D0: 48 49 2D 53 43 4F 52 45 09 39 39 39 39 39 20 20 ; 03A0
E0E0: 20 20 20 53 43 45 4E 45 09 39 39 39 20 20 20 20 ; 0302
E0F0: 20 4E 77 E4 3A EA EA EA 20 20 20 87 87 87 87 87 ; 07C4
E100: 87 87 87 87 87 C8 FE FF FF FF FF FF EF 8C FF FF ; 0CDD
E110: FF FF FF FF FF FF FF FF FF 7F 77 77 77 F7 FF FF ; 0DD0
E120: FF FF 00 00 00 00 00 FF FF 80 EC EE BF 99 FB EE ; 0997
E130: CE 08 00 33 E6 7F 22 77 22 F7 6E 33 00 01 03 08 ; 04CD
E140: 09 0C 0E 10 30 33 35 48 53 5B 6C 80 84 90 B5 C0 ; 0536
E150: C6 E0 EE 42 10 88 36 00 2C F6 F7 0C 00 79 C9 28 ; 0733
E160: 80 00 14 7B F3 E5 00 47 41 4D 45 20 4F 56 45 52 ; 055D
E170: 20 21 00 59 4F 55 20 47 4F 54 20 41 20 48 49 2D ; 0387
E180: 53 43 4F 52 45 20 21 21 00 33 01 2D 01 28 01 22 ; 028B
E190: 01 FF 02 33 01 2D 01 28 01 22 01 FF 02 00 2C 22 ; 02FF
E1A0: 04 00 00 2C 22 0C 00 00 00 0F 87 99 0F 00 00 87 ; 0223
E1B0: 88 07 00 00 00 0B 41 46 20 20 20 42 43 20 20 20 ; 0266
E1C0: 44 45 20 20 20 48 4C 20 20 20 49 58 20 20 20 49 ; 0327
E1D0: 59 20 20 20 50 43 20 20 20 53 50 02 01 48 1C C8 ; 037E
E1E0: 50 00 02 01 C8 50 00 48 20 49 20 54 20 20 20 4B ; 033B
E1F0: 20 45 20 59 20 20 20 21 20 21 20 20 20 20 20 20 ; 0260
E200: 47 20 41 20 4D 20 45 20 20 20 53 20 54 20 41 20 ; 0322
E210: 52 20 54 20 A5 20 A5 20 A5 20 46 20 55 20 4E 20 ; 047E
E220: 43 20 54 20 49 20 4F 20 4E 20 A5 20 35 00 20 47 ; 037E
E230: 20 41 20 4D 20 45 20 20 20 45 20 4E 20 44 20 A5 ; 036F
E240: 20 A5 20 A5 20 A5 20 A5 20 53 20 54 20 4F 20 50 ; 04DA
E250: 20 20 20 4B 20 45 20 59 00 B8 0A 58 50 58 50 00 ; 039B
E260: 1E 04 1F 05 20 06 21 07 22 07 23 07 24 07 25 07 ; 013E
E270: 26 07 27 07 28 06 29 05 2A 04 2B 03 2C 02 2D 01 ; 016F
E280: 00 2C 22 04 2E 22 0C 48 42 08 2C 22 04 2E 22 02 ; 01E4
E290: 94 99 06 1F 11 00 2F 22 0F 87 88 04 9F 99 08 20 ; 0436
E2A0: 2E 00 8E 00 0E 0E 00 0E 48 42 08 2E 42 08 2E 22 ; 0240
E2B0: 02 2E 22 0C 80 8F 00 0F 21 0F 43 48 03 2F 22 0F ; 029A
E2C0: 8F 48 03 9F 99 08 1F 53 08 62 79 20 4B 2E C2 B6 ; 0580
E2D0: BA DE BC 20 69 6E 20 46 4F 52 45 53 49 47 48 54 ; 0616
E2E0: 20 28 31 39 38 31 F2 20 31 31 F3 20 32 39 F4 29 ; 052A
E2F0: 00 98 28 E8 50 00 38 28 E8 50 00 4D 20 59 20 53 ; 04C9
E300: 20 54 20 45 20 52 20 59 00 58 D8 98 00 35 20 30 ; 0411
E310: 20 C3 20 DD 00 32 20 30 20 C3 20 DD 00 31 20 30 ; 04C3
E320: 20 C3 20 DD 00 4B 20 45 20 59 95 95 95 34 00 4B ; 0547
E330: 20 45 20 59 95 95 95 36 00 42 20 45 20 41 20 4D ; 0448
E340: 95 95 95 53 20 50 20 41 20 43 20 45 20 20 20 4B ; 0456
E350: 20 45 20 59 00 D8 0E 88 41 D8 00 B8 13 22 14 A8 ; 050E
E360: 3C 80 3D B8 50 00 31 20 35 20 30 20 30 20 C3 20 ; 042A
E370: DD 20 20 20 A6 20 20 20 BA 20 B4 20 D9 20 C4 CB ; 0679
E380: DE 20 B0 20 D1 20 CE 20 B3 20 20 20 B6 DE 20 00 ; 0674
E390: 31 20 C0 DE 20 B2 20 20 20 C2 20 B2 20 B6 20 BB ; 0666
E3A0: 20 DA 20 CF 20 BD 20 A1 20 00 03 FF 12 16 14 FF ; 05E4
```

マシン語学習の問題点を探る

マシン語入門セミナ開催記

「MCCクラブ活動報告」

はじめに

これから紹介するのは，埼玉県南部で活動しているマイコン・クラブMCC(マイコン・クラブ・クリエイタ）のドキュメントです。同好会からスタートした同クラブが，種々の葛藤の中から「マイコン教室」をスタートさせ，そしてその発展として「ＰＣ－8001マシン語セミナ」を開くまでの活動記録です。

このなかで，

①マイコン・クラブの問題点

②「ＰＣ－8001マシン語セミナ」ドキュメント

③ＰＣ－8001でマシン語を学ぶときの問題点とその解決

を中心に報告したいと思います。

- クラブ活動をしている人
- クラブ活動をしていない人
- これからどこかのクラブに加入しようと思っている人
- これからマシン語を勉強しようと思っている人

等の参考になれば幸いです。

3人寄れば「マイコン・クラブ・クリエイタ」誕生

埼玉県戸田市。

荒川を境に東京の隣りに位置するこの小さな市にも，御多聞にもれずマイコンの波が静かに押し寄せていた。

約２年前，戸田市の中町で家庭電気店を経営しているＳ氏を中心に熱心なマイコン・ファン(主にPC－8001の初期のユーザー）が集まり，熱い議論をかわしていた。当時はＰＣ－8001の出たての頃で，まだ使い方もあまり知られておらず，何とか動かそうと必死に情報交換をしていたそうです。その熱心さは凄まじいもので，ある人は雪の日でも１時間以上の距離から電車やバスを乗り継ぎ，ＰＣ－8001とモニタ・テレビをかかえて会場にかけつけたということです。

そして55年５月

正式に「マイコン・クラブ」としての

活動を開始し，

ここに「マイコン・クラブ・クリエイタ」の誕生となったわけです（第１表)。

54．10	●ＰＣ－8001を中心に同好会発足
55．5	●総会 ＝ クラブ誕生
	●マイコン・ショウ
	●(筆者ＰＣ－8001購入)
11	●マイコン教室（ＢＡＳＩＣ）スタート
56．3	●　〃　第１期終了
4	●「ＰＣ－8001マシン語入門セミナ」開催
5	●マイコン教室（ＢＡＳＩＣ）スタート
	●マイコン・ショウ

《第１表》ＭＣＣ略歴（資料：ＭＣＣ会員名簿）

マイコン・クラブの問題点

56年の10月，私はMCCの世話人であるＳ氏から突然，

「クラブでマイコン教室を開きたいので，講師をお願いしたい」

との要請を受けました。よく聞いてみると，
「クラブは発足したもののうまくいっていない。いつも情報交換の場で終わってしまう。情報交換ならまだよいが，何をやっているのかわからないうちに終ってしまう。そんな状態だからせっかく新会員が入ってきても，旧会員がやめてしまうので運営がうまくいかない」
と言うのです。S氏の話から，クラブ運営上の問題をまとめてみると，**第2表**のようになります。

つまり同じ趣味の同好会でありながら，マイコン・クラブには，

目的・志向の違い
機種による違い
技術の落差

という大きな問題点があるためまとまりにくく，しかも，「マイコン志向者はどちらかというと閉鎖的・利己的人間が多く，一層運営を困難にしている」と言うのです。すべてがそうであるというわけでもないのですが，たしかにそういう面はあると思います。そしてS氏が達した結論は次のとおりです。

「MCCマイコン教室」スタート

マイコン・ホビーストがクラブに求めるのは**第3表**のとおりです。これらを満たすことが，クラブ運営上必要になる。その解決策として

マイコン教室

を開く。その条件は**第4表**を守ること。つまり，
「ハードウエアによる差の出るキー・ワードは用いないで，ソフトを作る。レベルは最低線におくが，ソフトの質は落とさない。この条件を守った上でマイコン教室を開けば，機種の違い，レベルの違いを乗り越え，多くの者の希望を満たすことができるのではないだろうか？」
と考えたのです。そして，
「その講師をお願いしたい」
と言ってきたのです。びっくりした私はマイコンを始めて間もないし，まだ私自身がマイコンの勉強中であるので躊躇しました。しかしS氏の熱心さに，とうとう講師を引き受けることになってしまいました。こうして始まったのが，「MCCマイコン教室」なのです。

1．個人による目的意識の違い
(1)ハードウエア・ソフトウエア ｝志向による相違
(2)業務目的・ゲーム志向・システム研究・… ｝目的の相違

2．機種の違い
(1)ハードの違い
(2)ソフトの違い
- 同じBASICでも機種により異なり，同じに学習できない
- 作ったソフトの互換性がなく，他機種のソフトに興味をもてない

3．個人による技術の差が大きく，囲碁のようにハンディキャップがきかない
(1)上級者
- 大体入会の目的が技術の向上にあり，それが得られないと興味をしめさない
- 下級者と話をしても面白くない

(2)下級者
- クラブに入っても教えてくれる人がいない
- 上級者の話は難しくてわからず，話しても面白くない

《第2表》マイコン・クラブ運営上の問題点

1．技術指導者
2．ソフトウエア
……自分のマシンですぐ使えるもの
3．仲　間
……ハードウエア，ソフトの志向，技術レベルが一致していること

《第3表》マイコン・ホビーストがクラブに求めるもの

1．言語は全機種共通の標準BASICとする
2．そのためキー・ワードに制限を設ける
3．ソフトウエアを中心テーマにおく
4．レベルはキー・ボードのたたき方を知らない者におく

《第4表》マイコン教室の条件

さらに進んで

第1期「MCCマイコン教室(標準BASIC)」は，毎月第1日曜，合計5回にわたって開かれました。その間の出来事としては，

①初級者がクラブに顔を出すようになった。

②中学生から年輩者まで，クラブ員の層が広がった。

また講師の立場としては，

①自分ではマイコンを持っていない中学生が，クラブでマシンを動かしているうちに，プログラムを自作するようになった。

②標準BASICの範囲内で，構造化プログラミングの話をしたら，大学でFORTRANを学んだ人でさえ興味を示してくれたことは感激でした。

そこで私達は，せっかく教室がもり上がってきたのだから，ここらで，

マシン語のセミナ

を開いてみよう，ということになりました。マシン語のセミナは他ではあまり見当たらない。どうせやるなら外部の人にも聞いてもらおう，ということで，

「PC－8001マシン語入門セミナ」

を企画することになったのです。

方　針

セミナは外部の人も招くので，1日コースに決まりました。

私は初め

「どうせ1日コースだ。BASICだって教えるには何日もかかる。ましてマシン語だ。1日じゃたいしたことは出来ない。簡単な命令だけ教えて，お茶をにごそう」

という方針に決めました。

ところが——。

その方針を変えざるを得なくなったのです。

反　響

「PC－8001マシン語入門セミナ」の開催を計画してびっくりしました。予想以上に反響が大きかったのです。BASIC教室には，見向きもしなかった上級クラブ員が，多数申し込んできたのです。また外部からの申し込みも多く，予約しておいた会場の定員を軽くオーバーしてしまいました。

申し込み者のハガキを見ると，

「私の地区ではこういう催しを開くチャンスがありません。参加したい」

というものが圧倒的に多かった。つまり，

BASICではなく

「マシン語セミナ」への期待

が予想以上に大きかったのです。

さらにつけ加えれば，申し込み者の中には何と福島県の人が含まれていました。私達は，

「福島県からでは無理だから」

と一度はお断わりしたのですが，その熱心さについに断わりきることができませんでした。

以上のように，申し込み者の期待は大きかった，そしてだんだんと，

「これはただじゃ済みそうもない」

と思われてきました。

「簡単な命令を教えるだけでは申し分けない」

ということになり，当初の方針を変更する気になりました(**第1図**)。

限定解除

さて問題です。

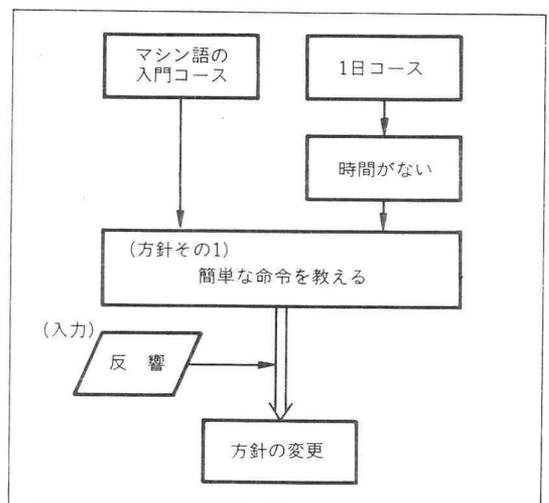
①1日コースで，時間は限定されている。

②相手は入門者

の壁を乗り越えて，マシン語をマスターしてもらおうとするのです。

そこで私の採用した方針は次のとおりです。

まず相手が入門者だからと言って，

やさしい命令に限定————————(×)

《第1図》「マシン語セミナの方針」流れ図

する方針はさけました。そして最初から,

一つのまとまったプログラムに挑戦——(○)

する方法をとることにしました。**リスト1**を見てくだ

```
E08F 011950      LD   BC,5019H
E092 CD3A09      CALL 093AH
E095 211801      LD   HL,0118H
E098 225DEA      LD   (0EA5DH),HL
E09B 01FF00      LD   BC,00FFH
E09E CDF708      CALL 08F7H
E0A1 CD5A04      CALL 045AH
E0A4 3E87        LD   A,87H
E0A6 0604        LD   B,04H
E0A8 117800      LD   DE,0078H
E0AB 21C1F6      LD   HL,0F6C1H
E0AE 77          LD   (HL),A
E0AF 19          ADD  HL,DE
E0B0 10FC        DJNZ 0E0AEH
E0B2 3EE5        LD   A,0E5H
E0B4 323AF7      LD   (0F73AH),A
E0B7 3EE7        LD   A,0E7H
E0B9 32B2F7      LD   (0F7B2H),A
E0BC 1110F7      LD   DE,0F710H
E0BF 0604        LD   B,04H
E0C1 C5          PUSH BC
E0C2 D5          PUSH DE
E0C3 CDF3E0      CALL 0E0F3H
E0C6 D1          POP  DE
E0C7 C1          POP  BC
E0C8 7B          LD   A,E
E0C9 C678        ADD  A,78H
E0CB 5F          LD   E,A
E0CC 7A          LD   A,D
E0CD CE00        ADC  A,00H
E0CF 57          LD   D,A
E0D0 10EF        DJNZ 0E0C1H
E0D2 213BF7      LD   HL,0F73BH
E0D5 3681        LD   (HL),81H
E0D7 3600        LD   (HL),00H
E0D9 23          INC  HL
E0DA 7D          LD   A,L
E0DB FE88        CP   88H
E0DD 3803        JR   C,0E0E2H
E0DF 213BF7      LD   HL,0F73BH
E0E2 3681        LD   (HL),81H
E0E4 CDF10C      CALL 0CF1H
E0E7 3807        JR   C,0E0F0H
E0E9 0605        LD   B,05H
E0EB CDFCE0      CALL 0E0FCH
E0EE 18E7        JR   0E0D7H
E0F0 C3665C      JP   5C66H
E0F3 010700      LD   BC,0007H
E0F6 2109E1      LD   HL,0E109H
E0F9 EDB0        LDIR
E0FB C9          RET
E0FC CD01E1      CALL 0E101H
E0FF 10FB        DJNZ 0E0FCH
E101 D9          EXX
E102 16FF        LD   D,0FFH
E104 15          DEC  D
E105 20FD        JR   NZ,0E104H
E107 D9          EXX
E108 C9          RET
E109 000001C8    DB   00H,00H,01H,0C8H
E10D 03A850      DB   03H,0A8H,50H
```

《リスト1》マシン語セミナMAIN教材(やさしい命令だけでなくDJNZ, LDIR等の命令も含まれる。GE08F↗で走らせてみてください。

さい。これが「マシン語セミナ」で挑戦したMAIN教材です。

JR, DJNZ ————相対番地計算必要
LD, IR—————ブロック転送

の命令等,初心者にはちょっと,とりつきにくいものさえあえてさけなかったのです。しかも短いプログラムながら,

①メーカー非公開のROM内サブルーチンの使用

②画面の初期設定をマシン語で行なう

③色を変えるため,マシン語でアトリビュート・エリアを直接いじる。

等,本格的なものです。これらをすべて初心者にやってもらったのです。これはBASICをマスターした上級者でもちょっときついのではないでしょうか? こんなことを初心者に押しつけることが可能なのでしょうか? しかし,実際私はこの方針を強行しました。

そして,その結果は——?

反 応

「まったくの初心者です。マシン語の良いステップになりました」

「8080時代のマシン語の経験はおおむね亡却してしまったがいくらかよみがえり,新たにPC—8001でマシン語に挑戦する勇気が出た」

「家に帰るのが楽しみデス」

「アッ!! というまに時間が過ぎてしまって残念です。少しだけですけどわかることができたみたいで満足でございます。又,こんな機会を作ってください」

「本講習に参加した目的の一つは,PC—8001のメモリ・マップ,システム・サブルーチンを知りたいことだった。これは資料(後述)により満足できた」

以上は「セミナ」修了後にとったアンケートから,ランダムに抜き出したものです。アンケートを取ってみてわかったことは,「マシン語入門セミナ」とうたったにもかかわらず,初級者の他に中級者,上級者も含まれていたということ。そしてそれぞれの反応を分析すると,次のとおりです。

①初級者

「セミナ」参加者の大部分はマシン語はおろか,マイコンそのものの初心者でした。彼等の反応は,「聞いているときはわかるが自分で応用するのは無理です。そして同レベルのものをもう1度聞いてみ

たい」というのが多かったようです。

②中級者

彼等はBASICの大家ではあるが，マシン語の挫折組である。しかし「セミナ」での彼等の理解度は80%以上であり，これをきっかけにさらに進んでいこうとする意欲がうかがわれました。

③上級者

彼等の大部分はワン・ボード・マイコンの経験者である。したがって少なくとも8080の命令は理解できる人達であり，マイコンの世界では私の大先輩にあたる。彼等の「セミナ」参加の目的は，非公開であるPC－8001の中身を知りたくてやって来たようです。

以上のように「セミナ」参加者には，初級から上級の人まで含まれていた。そして私のやり方はそれぞれのレベルの人に，それなりに少しは満足していただけたように思われます。(私の独断と偏見で)。

クラブ運営上の問題から出発して，MCCでは「マシン語セミナ」を開くことになりました。ところが反響の大きさに，内容の変更をせざるを得なくなったわけです。

時　分　割

さて，中・上級者でもある程度興味を示してくれるレベルの教材を，まったくの初心者に，しかも１日のコースでいかに消化してもらったのでしょうか？　以下に，いよいよ「PC－8001マシン語セミナ」のドキュメントを御紹介しましょう。

前述の**リスト１**が「セミナ」でのMAINの教材です。これをマシン語の初心者にいきなりぶっつけるのは，いくらなんでもちょっと無理です。そこで私は１日のコースを，**第５表**のようにおおむね三つのパートに分けました。

パート３がメインで，**リスト１**のMAIN教材の解説と実験にあてました。パート２がその基礎知識を与えるもので，この時間を使って

Z-80命令表の見方

マシン語プログラムの書き方

を理解してもらいました。パート１はさらにその基礎となる部分で，マシン語の世界への導入に当てたわけです。

それではだんだんと核心にせまっていきましょう。

《第５表》PC－8001マシン語入門セミナ（記録）

①セミナの内容

ＰＣ－8001のモニタの使い方

Ｚ－80命令表の使い方

他

②開催日：81年５月５日（にやりました！）

④時　間：

1時間目	10：00～10：50	パート１
2　〃	11：10～12：00	パート２
3　〃	1：00～ 2：20	パート３
4　〃	2：40～ 4：00	

《第６表》PC－8001↔MZ－80

	ＰＣ－8001	ＭＺ－80
カラー	○	×
音楽演奏	×	○

PC-8001対MZ-80

現在マイコンを代表する機種といえば，

ＰＣ－8001

ＭＺ－80

の二つでしょう。雑誌に紹介されるプログラムを見ても，80%以上は上の二つに限定されています。

私もマイコンを購入するにあたって，上の二つのどちらを選ぶか迷いました。選択にあたっての決定的な違いは，**第６表**のようでしょう。

つまり，

カラーをとるか——ＰＣ－8001

音楽をとるか——ＭＺ－80

の選択になるわけです。ところが——。

PC-8001による音楽演奏

私は以前，ＴＫ－80による音楽演奏の記事を読んだことがありました。だからマシン語を使えば，

PC－8001でも音楽演奏が可能では？

と思ったのです。そこで56年の５月末，「マイコン・ショウ」で新機種が出ていないことを確認すると，すぐにPC－8001を購入しました。

最初は内蔵スピーカーの制御がわからなかったのですが，56年７月，ついに成功しました。

PC−8001による音楽演奏に！

もっとも残念ながらマイコン誌「マシン・コード」連載中の川村さんが，私より先に音楽演奏に成功していましたが。

導　入

1時間目の導入部に，私は以上のような話から始めました。そしてFORESIGHTの川村さんの作った「エリーゼのために」を実際に聞いてもらいました。

つまり，

「BASICもいいでしょう。でもPC−8001の場合，BASICでは演奏できません。それは不可能というものです。しかし，PC−8001はもともと音楽の演奏をやろうと思えば，その実力を持っているのです。ただ，BASICではその性能を引き出すことができないだけです。

マシン語を学んでみてください！

マシン語を使えば，BASICでは引き出せないPC−8001の本当の実力を，あなたが直接引き出すことができるのです。学んでください。学びましょう。マシン語を！」

ということを強調したかったのです。

「マシン語って，すごいな，すごいな。マシン語を学んでみたいな」

という気にさせたかったのです。興味を持ってくれれば，マシン語への動機付けができれば，あとは比較的スムーズに事が運ぶでしょう。

実　験　1

それでは実際にあなたにもマシン語を経験していただきましょう。

もしあなたが，まだPC−8001による音

```
D000 AF          XOR  A
D001 D351        OUT  51H,A
D003 CD10D0      CALL 0D010H
D006 011928      LD   BC,2819H
D009 CD50D0      CALL 0D050H
D00C CD3A09      CALL 093AH
D00F 76          HALT
D010 11FF35      LD   DE,35FFH
D013 2162D1      LD   HL,0D162H
D016 CD93D0      CALL 0D093H
D019 2135D1      LD   HL,0D135H
D01C CD93D0      CALL 0D093H
D01F 21BBD1      LD   HL,0D1BBH
D022 CD93D0      CALL 0D093H
D025 21FED1      LD   HL,0D1FEH
D028 CD93D0      CALL 0D093H
D02B 2115D2      LD   HL,0D215H
D02E CD93D0      CALL 0D093H
D031 C9          RET
D032 11FF20      LD   DE,20FFH
D035 21BBD1      LD   HL,0D1BBH
D038 CD93D0      CALL 0D093H
D03B C9          RET
D03C 11FF25      LD   DE,25FFH
D03F 21FED1      LD   HL,0D1FEH
D042 CD93D0      CALL 0D093H
D045 C9          RET
D046 11FF20      LD   DE,20FFH
D049 2115D2      LD   HL,0D215H
D04C CD93D0      CALL 0D093H
D04F C9          RET
D050 160A        LD   D,0AH
D052 014020      LD   BC,2040H
D055 CD6AD0      CALL 0D06AH
D058 15          DEC  D
D059 20F7        JR   NZ,0D052H
D05B C9          RET
D05C F5          PUSH AF
D05D 3EFF        LD   A,0FFH
D05F CD7AD0      CALL 0D07AH
D062 3D          DEC  A
D063 20FA        JR   NZ,0D05FH
D065 0D          DEC  C
D066 20F5        JR   NZ,0D05DH
D068 F1          POP  AF
D069 C9          RET
D06A CD7AD0      CALL 0D07AH
D06D 04          INC  B
D06E 0D          DEC  C
D06F 20F9        JR   NZ,0D06AH
D071 C9          RET
D072 CD7AD0      CALL 0D07AH
D075 05          DEC  B
D076 0D          DEC  C
D077 20F9        JR   NZ,0D072H
D079 C9          RET
D07A D5          PUSH DE
D07B 50          LD   D,B
D07C 3A67EA      LD   A,(0EA67H)
D07F CBEF        SET  5,A
D081 D340        OUT  40H,A
D083 15          DEC  D
D084 20FB        JR   NZ,0D081H
D086 50          LD   D,B
D087 3A67EA      LD   A,(0EA67H)
D08A CBAF        RES  5,A
D08C D340        OUT  40H,A
D08E 15          DEC  D
D08F 20FB        JR   NZ,0D08CH
D091 D1          POP  DE
D092 C9          RET
D093 7E          LD   A,(HL)
D094 A7          AND  A
D095 C8          RET  Z
D096 47          LD   B,A
D097 23          INC  HL
D098 4E          LD   C,(HL)
D099 07          RLCA
D09A 3005        JR   NC,0D0A1H
D09C CDB4D0      CALL 0D0B4H
D09F 1803        JR   0D0A4H
D0A1 CDE9D0      CALL 0D0E9H
D0A4 3A67EA      LD   A,(0EA67H)
D0A7 CBAF        RES  5,A
D0A9 D340        OUT  40H,A
D0AB 0610        LD   B,10H
D0AD CD2DD1      CALL 0D12DH
D0B0 10FB        DJNZ 0D0ADH
D0B2 18DF        JR   0D093H
D0B4 E5          PUSH HL
D0B5 D5          PUSH DE
D0B6 CDC0D0      CALL 0D0C0H
D0B9 D1          POP  DE
D0BA 0D          DEC  C
D0BB 20F8        JR   NZ,0D0B5H
D0BD E1          POP  HL
D0BE 23          INC  HL
D0BF C9          RET
D0C0 EB          EX   DE,HL
D0C1 50          LD   D,B
D0C2 3A67EA      LD   A,(0EA67H)
D0C5 CBAF        RES  5,A
D0C7 D340        OUT  40H,A
D0C9 2B          DEC  HL
D0CA 7C          LD   A,H
D0CB A7          AND  A
D0CC 2815        JR   Z,0D0E3H
D0CE 15          DEC  D
D0CF 20F8        JR   NZ,0D0C9H
D0D1 50          LD   D,B
D0D2 3A67EA      LD   A,(0EA67H)
D0D5 CBAF        RES  5,A
D0D7 D340        OUT  40H,A
D0D9 2B          DEC  HL
D0DA 7C          LD   A,H
D0DB A7          AND  A
D0DC 2805        JR   Z,0D0E3H
D0DE 15          DEC  D
D0DF 20F8        JR   NZ,0D0D9H
D0E1 18DE        JR   0D0C1H
D0E3 3A67EA      LD   A,(0EA67H)
D0E6 CBAF        RES  5,A
D0E8 C9          RET
D0E9 E5          PUSH HL
D0EA D5          PUSH DE
D0EB CDF5D0      CALL 0D0F5H
D0EE D1          POP  DE
D0EF 0D          DEC  C
D0F0 20F8        JR   NZ,0D0EAH
D0F2 E1          POP  HL
D0F3 23          INC  HL
D0F4 C9          RET
D0F5 EB          EX   DE,HL
D0F6 50          LD   D,B
D0F7 3A67EA      LD   A,(0EA67H)
D0FA CBEF        SET  5,A
D0FC D340        OUT  40H,A
D0FE 2B          DEC  HL
D0FF 7C          LD   A,H
D100 A7          AND  A
D101 2815        JR   Z,0D118H
D103 15          DEC  D
D104 20F8        JR   NZ,0D0FEH
D106 50          LD   D,B
D107 3A67EA      LD   A,(0EA67H)
D10A CBAF        RES  5,A
D10C D340        OUT  40H,A
```

《リスト2》PC−8001による音楽演奏（次頁に続く）

```
D10E 2B         DEC  HL
D10F 7C         LD   A,H
D110 A7         AND  A
D111 2805       JR   Z,0D118H
D113 15         DEC  D
D114 20F8       JR   NZ,0D10EH
D116 18DE       JR   0D0F6H
D118 C9         RET
D119 CD2DD1     CALL 0D12DH
D11C 10FB       DJNZ 0D119H
D11E C9         RET
D11F D5         PUSH DE
D120 11FFFF     LD   DE,0FFFFH
D123 1D         DEC  E
D124 C223D1     JP   NZ,0D123H
D127 15         DEC  D
D128 C223D1     JP   NZ,0D123H
D12B D1         POP  DE
D12C C9         RET
D12D D5         PUSH DE
D12E 1EFF       LD   E,0FFH
D130 1D         DEC  E
D131 20FD       JR   NZ,0D130H
D133 D1         POP  DE
D134 C9         RET
D135 3303FF01   DB   33H,03H,0FFH,01H
D139 3303FF01   DB   33H,03H,0FFH,01H
D13D 33042D06   DB   33H,04H,2DH,06H
D141 28022D04   DB   28H,02H,2DH,04H
D145 28042804   DB   28H,04H,28H,04H
D149 2604220C   DB   26H,04H,22H,0CH
D14D 26042204   DB   26H,04H,22H,04H
D151 1E042806   DB   1EH,04H,28H,06H
D155 26022804   DB   26H,02H,28H,04H
D159 2D042D04   DB   2DH,04H,2DH,04H
D15D 36043308   DB   36H,04H,33H,08H
D161 00220222   DB   00H,22H,02H,22H
D165 021E021E   DB   02H,1EH,02H,1EH
D169 02220326   DB   02H,22H,03H,26H


D16D 01280422   DB   01H,28H,04H,22H
D171 0222021E   DB   02H,22H,02H,1EH
D175 021E0222   DB   02H,1EH,02H,22H
D179 03260128   DB   03H,26H,01H,28H
D17D 04220222   DB   04H,22H,02H,22H
D181 021E021B   DB   02H,1EH,02H,1BH
D185 02190416   DB   02H,19H,04H,16H
D189 041B041B   DB   04H,1BH,04H,1BH
D18D 011E011B   DB   01H,1EH,01H,1BH
D191 011E0122   DB   01H,1EH,01H,22H
D195 041E021B   DB   04H,1EH,02H,1BH
D199 02190419   DB   02H,19H,04H,19H
D19D 021E0222   DB   02H,1EH,02H,22H
D1A1 081E0422   DB   08H,1EH,04H,22H
D1A5 02260228   DB   02H,26H,02H,28H
D1A9 08220222   DB   08H,22H,02H,22H
D1AD 02330226   DB   02H,33H,02H,26H
D1B1 0228042D   DB   02H,28H,04H,2DH
D1B5 04330EFF   DB   04H,33H,0EH,0FFH
D1B9 02002204   DB   02H,00H,22H,04H
D1BD 28032601   DB   28H,03H,26H,01H
D1C1 22043304   DB   22H,04H,33H,04H
D1C5 2D012801   DB   2DH,01H,28H,01H
D1C9 26012201   DB   26H,01H,22H,01H
D1CD 26022802   DB   26H,02H,28H,02H
D1D1 2D014401   DB   2DH,01H,44H,01H
D1D5 44013D01   DB   44H,01H,3DH,01H
D1D9 44014C01   DB   44H,01H,4CH,01H
D1DD 51015B01   DB   51H,01H,5BH,01H
D1E1 28012601   DB   28H,01H,26H,01H
D1E5 22011E01   DB   22H,01H,1EH,01H
D1E9 22022202   DB   22H,02H,22H,02H
D1ED 19042204   DB   19H,04H,22H,04H
D1F1 26022801   DB   26H,02H,28H,01H
D1F5 2D012D03   DB   2DH,01H,2DH,03H
D1F9 33013306   DB   33H,01H,33H,06H
D1FD 001E041E   DB   00H,1EH,04H,1EH
D201 031E011E   DB   03H,1EH,01H,1EH
D205 0419031B   DB   04H,19H,03H,1BH
D209 011B031E   DB   01H,1BH,03H,1EH
D20D 011E0320   DB   01H,1EH,03H,20H
D211 011E0800   DB   01H,1EH,08H,00H
D215 33012D01   DB   33H,01H,2DH,01H
D219 28012202   DB   28H,01H,22H,02H
D21D 28012204   DB   28H,01H,22H,04H
D221 00         DB   00H
```

音楽演奏を聞いたことがなければ，

リスト２のマシン語

(D000－D221)

を打ち込んで，

GD000↗

で走らせてください。標準装備のＰＣ－8001でも，いろいろな音楽や電子音を楽しむことができることがわかるでしょう。

実 験 2

以上のようなマシン語でのみ可能な実験を，導入部のみならず，セミナの随所に設けました。これは興味を持続させるためで，他意はありません。もう少しその実験を紹介しておきましょう。

(リスト3) E000－E04F

を打ち込んでください。次の一つの実験をします。

1. GE000↗

これを実行して画面を見ると，何か変わった感じがしますね。次の２点のイタズラをしました。

①ファンクション・キーの内容表示がリバースされていない。

②ファンクション・キーのf･6－f･10の内容が表示されている。

2. GE015↗

画面の上部に数字の列が表示されましたが(**第2図**)，これ，凄いと思いませんか？ サッと見ると単なる数字の羅列です。0－9までの数が，白・黄・シアン・緑色で表示されています。ということはもちろんカラーですね。ちょっと待ってください。

①字数をもう一度数えてみてください。40字あります。しかもカラー・モードです。

気付きましたか？ 驚異ですね。つまりカラー・モードでは39字しか表示されないはずなのに，40字表示されているのです。

②黄色の１がブリンクしています。緑の2・3がリバース・ブリンクしてい

```
E000 AF           XOR  A
E001 325CEA       LD   (0EA5CH),A
E004 CD5A04       CALL 045AH
E007 3E05         LD   A,05H
E009 CDC907       CALL 07C9H
E00C 01FFFF       LD   BC,0FFFFH
E00F CDF708       CALL 08F7H
E012 C3665C       JP   5C66H
E015 2100F3       LD   HL,0F300H
E018 0604         LD   B,04H
E01A 0E0A         LD   C,0AH
E01C 3E30         LD   A,30H
E01E 77           LD   (HL),A
E01F 3C           INC  A
E020 23           INC  HL
E021 23           INC  HL
E022 0D           DEC  C
E023 20F9         JR   NZ,0E01EH
E025 10F3         DJNZ 0E01AH
E027 011B00       LD   BC,001BH
E02A 1150F3       LD   DE,0F350H
E02D 2135E0       LD   HL,0E035H
E030 EDB0         LDIR
E032 C3665C       JP   5C66H
E035 000002E8     DB   00H,00H,02H,0E8H
E039 14C81602     DB   14H,0C8H,16H,02H
E03D 18001E20     DB   18H,00H,1EH,20H
E041 28A82A10     DB   28H,0A8H,2AH,10H
E045 32003C88     DB   32H,00H,3CH,88H
E049 40064400     DB   40H,06H,44H,00H
E04D 483050       DB   48H,30H,50H
```

《リスト3》マシン語の実験(イタズラ)

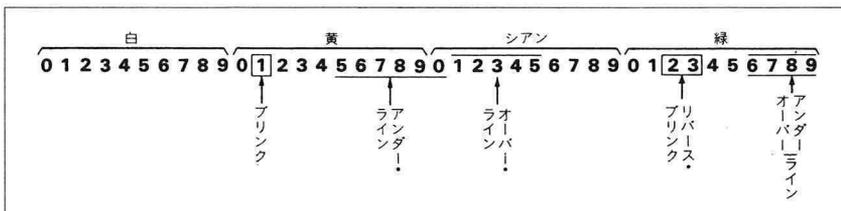
```
E050 76           HALT
E051 3E01         LD   A,01H
E053 0602         LD   B,02H
E055 0E03         LD   C,03H
E057 1604         LD   D,04H
E059 1E05         LD   E,05H
E05B 2606         LD   H,06H
E05D 2E07         LD   L,07H
E05F DD210800     LD   IX,0008H
E063 FD210900     LD   IY,0009H
E067 FF           RST  38H
E068 010302       LD   BC,0203H
E06B 78           LD   A,B
E06C 81           ADD  A,C
E06D FF           RST  38H
E06E DD2189E0     LD   IX,0E089H
E072 DD4600       LD   B,(IX+00H)
E075 DD7E02       LD   A,(IX+02H)
E078 80           ADD  A,B
E079 DD7704       LD   (IX+04H),A
E07C DD4601       LD   B,(IX+01H)
E07F DD7E03       LD   A,(IX+03H)
E082 88           ADC  A,B
E083 DD7705       LD   (IX+05H),A
E086 C3665C       JP   5C66H
E089 FE00         CP   00H
E08B 02           LD   (BC),A
E08C 00           NOP
E08D 00           NOP
E08E 00           NOP
E08F 00           NOP
```

《リスト4》マシン語セミナ基礎教材

ます。これはどうですか？BASICでもLINE文を使えばできますね――本当ですか？ カラー・モードでは1行の中の特定の文字だけブリンクさせることはできないはずです。

③黄色の5からシアンの0にかけて，アンダー・ラインが引いてあります。シアンの1から5にはオーバー・ラインが引いてあります。さらに緑の6から9には，御丁寧にアンダー・ラインとオーバー・ラインの2本が引いてあります。

これは一目瞭然，N-BASICでは無理です。でもあなたのPC-8001には，その能力が備わっています。

以上，マシン語を使ってイタズラをしてみました。いかがですか？ 先を急ぎましょう。

《第2図》カラー・モードでの驚異

HALTもいいが……

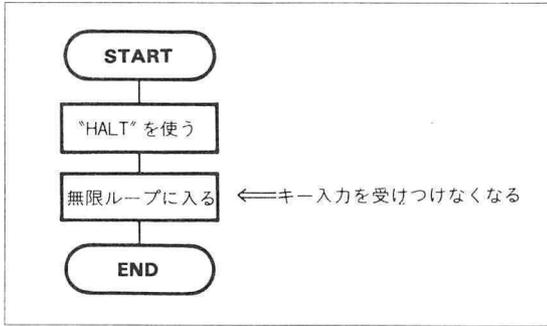
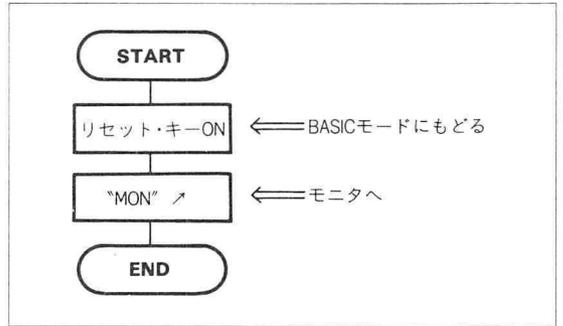
リスト4が，MAIN教材に入る前に使った教材です。ここでは次の4点がポイントです。

①E050

これはHALT命令を知ってもらうため。プログラムを停止させるためには不可欠なので，1番最初にもってきました。ところが――

実際にGE050↗で走らせると，確かにプログラムは実行を停止します。しかしその後は，どのキーを押しても(ストップ・キーでも)，ウンともスンとも言わなくなります。これは無限ループに入ってしまったからで，これを止めるにはリセット・キーしかありません(**第3図**，**第4図**)。

プログラムを走らせる度にこんな手間をかけるのはめんどうです。

以上のわずらわしさを体験してもらったあと，

《第3図》プログラムの実行を停止する

マシン語のプログラムを止めるには

HALT――――――――(×)

JP 5C66H ――――――(○)

の方法を示しました。つまりモニタのスタート・アドレスに飛ばせる方法です。これなら一発でプログラムを止めたあと，モニタ・コマンド・モードにもどります。

②E051－E067

LD命令で，レジスタに数値を入れる方法を覚える。

③E068－E06D

レジスタ・ペアの使い方。

④E06E－E08E

インデックス・レジスタの使い方と，2バイト加算によるフラグの変化について。

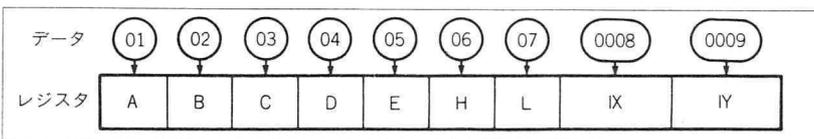
②－④については，以上がポイントです。実はこの中に，「PC－8001でマシン語を扱うための困難その1」が隠されているのです。次に②を例にしてそれを見ていくことにしましょう。

困難その1

いま第7表のマシン語を入力して実行させたとします。すると各レジスタに第5図のようなデータが入るはずです。そこで実際に

GE051↗

で走らせてみましょう。何が起こるでしょうか？ 何も

《第4図》無限ループから抜ける

E051	3E 01	LD A，01H
53	06 02	LD B，02H
55	0E 03	LD C，03H
57	16 04	LD D，04H
59	1E 05	LD E，05H
5B	26 06	LD H，06H
5D	2E 07	LD L，07H
5F	DD 21 08 00	LD IX，0008H
63	FD 21 09 00	LD IY，0009H
67	C3 66 5C	JP 5C66H

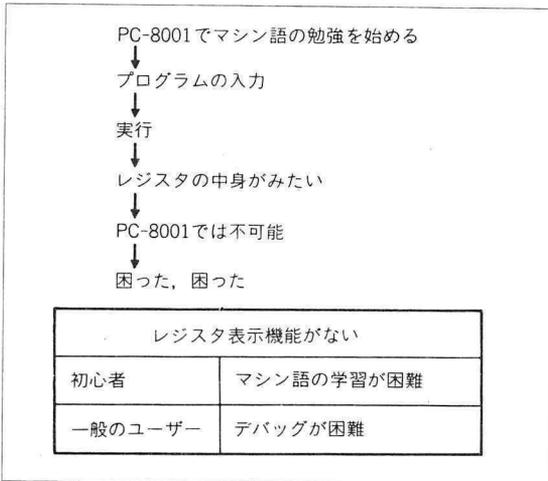
《第7表》レジスタに数を入れてみたが……

起こりません。＊が表示されてコマンド待ちになるだけです。それではレジスタの値を確かめるにはどうしたら良いのでしょうか？

答は，「無理です。理由はPC－8001にはレジスタ表示の機能がついていないから」ということになります。つまりまとめると第6図のようになるわけです。もともとPC－8001はマシン語での使用を想定しておらず，まして PC を使ってマシン語を学んでもらおうと意図していないので，モニタに最低必要限の機能しかついていないためです。それでは，その困難を克服するには？

レジスタの内容を表示させるプログラムを自作すればよいのです。しかしこれからマシン語を学ぼうとする人にはちょっと大変でしょう。そんなことが出来る位なら，マシン語を学ぶ必要はないわけですから。

そこで私はセミナで自分の作った「レジスタ表示プログラム」を公開し，その使い方だけを覚えてもらうことにしました。こうすることにより，**困難その1を克服**することができるわけです。それでは次にそのプログラムの使い方をみてみましょう。

《第5図》レジスタの中身は・・・・

《第 6 図》 困難その 1

「レジスタ表示プログラム」の使い方

このプログラム（**リスト 5**）の特徴は，

①完全リロケータブルである。したがってどの番地に置いてもかまわない。

②全体が短いので使いやすい（たとえば大きいプログラムを作るときには，これをサブルーチンとして入れておけば，デバッグに便利）。

使い方は，

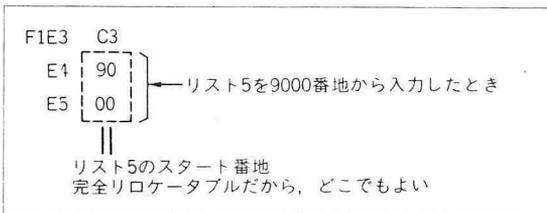
①任意の番地にプログラムを入力する。

②F 1 E 3 番地から 3 バイトを**第 7 図**のように変える。

③自分のプログラムで，レジスタの値を見たいところに RST 38H（マシン・コード＝FF） を書き込む。

あとは自分のプログラムを実行させるだけです。たとえば**第 7 表**のプログラムの場合，E067番地でレジスタの内容を見たいわけですから，ここを FF に変える。変えたものが**リスト 4**です。こうして

GE051↗

でプログラムを走らせれば，今度は**第 8 図**のようにレジスタの内容が表示されます。

《第 7 図》 F1E～F1E5を変える

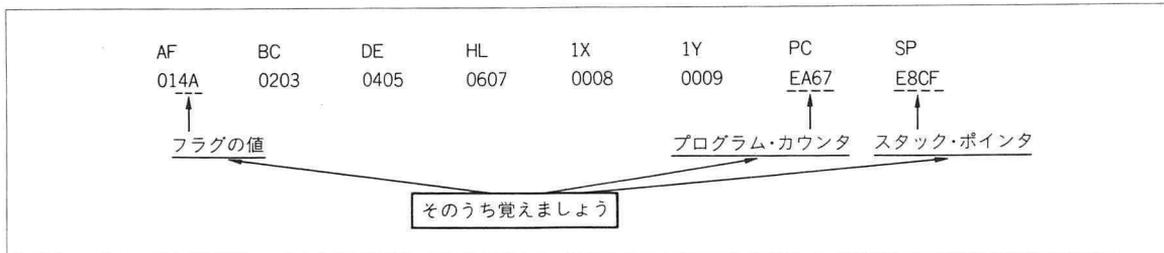
```
E110 FDE5      PUSH IY
E112 DDE5      PUSH IX
E114 E5        PUSH HL
E115 D5        PUSH DE
E116 C5        PUSH BC
E117 F5        PUSH AF
E118 CDCA5F    CALL 5FCAH
E11B 0625      LD   B,25H
E11D 210000    LD   HL,0000H
E120 39        ADD  HL,SP
E121 2B        DEC  HL
E122 2B        DEC  HL
E123 F9        LD   SP,HL
E124 E1        POP  HL
E125 113500    LD   DE,0035H
E128 19        ADD  HL,DE
E129 7E        LD   A,(HL)
E12A CD5702    CALL 0257H
E12D 23        INC  HL
E12E 10F9      DJNZ 0E129H
E130 CDCA5F    CALL 5FCAH
E133 0606      LD   B,06H
E135 E1        POP  HL
E136 CDC05E    CALL 5EC0H
E139 CDD45F    CALL 5FD4H
E13C 10F7      DJNZ 0E135H
E13E E1        POP  HL
E13F 2B        DEC  HL
E140 CDC05E    CALL 5EC0H
E143 CDD45F    CALL 5FD4H
E146 210000    LD   HL,0000H
E149 39        ADD  HL,SP
E14A CDC05E    CALL 5EC0H
E14D C3665C    JP   5C66H
E150 41462020  DB   41H,46H,20H,20H
E154 20424320  DB   20H,42H,43H,20H
E158 20204445  DB   20H,20H,44H,45H
E15C 20202048  DB   20H,20H,20H,48H
E160 4C202020  DB   4CH,20H,20H,20H
E164 49582020  DB   49H,58H,20H,20H
E168 20495920  DB   20H,49H,59H,20H
E16C 20205043  DB   20H,20H,50H,43H
E170 20202053  DB   20H,20H,20H,53H
E174 50        DB   50H
```

《リスト 5》 レジスタ表示プログラム

困難その 2

いよいよ最後になりました困難その 2 です。

PC－8001でレジスタの内容を見ることができるようになりました。そしてあなたが，完全にＺ－80の命令を使いこなせるようになったとします。そうすれば，PC－8001でマシン語が使えるでしょう——。

否，おそらく難しいのではないでしょうか。なぜなら，マシン語はDATAを直接周辺機器とやり取りするという性格を持っているため，個々のマシンによる扱い方が異なるのです。そして残念なことにPC－8001ではそのへんの中身がまったく公開されていないのです。たとえばマシン語でプリンタをいじりたいと思っても，その中身がわからないと，まったく手が出ない

《第8図》実行結果

1. Z－80命令活用表
2. 機械語↔ニーモニック対応表
3. スクリーン・エリア，アトリビュート・エリアの操作法
4. N－BASIC 1行の構成
5. ファンクション・キーの内部
6. RST命令活用法
7. システム・サブルーチン
(アドレス，使用法，変化するレジスタ等約70)
8. ワーク・エリア(約40)
9. データ・エリア

《第8表》「PC－8001マシン語活用マニュアル」の内容

でしょう。

この困難を克服するには，1日のセミナでは時間が足りません。そこでセミナ用の資料を用意することにしました。**第8表**がそれです。これは前述の川村さんと一緒にまとめたもので，PC－8001でマシン語を扱うのに困らないように内部のシステム・サブルーチンの使い方等を整理したものです。

セミナではこの資料の使い方を説明し，あとで必要になった時，随時利用してもらうことにしました。

おわりに

以上MCCの活動記録を通して，クラブ運営上の問題点，マシン語を学ぶときの問題点等に焦点を合わせてみました。いかがでしたか？ 皆さんのクラブについても，報告していただけませんか？ 参考にしたいと思います。

なお本稿は活動記録を主目的にしましたので，例にあげた教材については説明を省略しました。また最後に述べたPC－8001の中身については，「マイコン」誌や各マイコン誌に時々掲載されているようですから，まとめておくとよいでしょう。最後に，本誌以外でマシン語を勉強するための参考書を紹介しておきます。

① 「Z－80マイコン・プラグラミングテクニック」(電波新聞社)
Z－80の命令が丁寧に解説されている。
使用機種はTRS－80。

② 「マイコン機械語入門」(電波新聞社)
ソフト中心に興味深く読める。
使用機種はMZ－80。

《付録1-ⓐ》Z-80活用表

8 ビット

×	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX+d)	(IY+d)	n	(nn)	I	R
LD A, ×	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3E n	3A n n	ED 57	ED 5F
LD B, ×	47	40	41	42	43	44	45	46			DD 46 d	FD 46 d	06 n			
LD C, ×	4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d	0E n			
LD D, ×	57	50	51	52	53	54	55	56			DD 56 d	FD 56 d	16 n			
LD E, ×	5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d	1E n			
LD H, ×	67	60	61	62	63	64	65	66			DD 66 d	FD 66 d	26 n			
LD L, ×	6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d	2E n			
LD (HL), ×	77	70	71	72	73	74	75						36 n			
LD (BC), ×	02															
LD (DE), ×	12															
LD (IX+d), ×	DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d						DD 36 d n			
LD (IY+d), ×	FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d						FD 36 d n			
LD (nn), ×	32 n n															
LD I, ×	ED 47															
LD R, ×	ED 4F															
ADD A, ×	87	80	81	82	83	84	85	86			DD 86 d	FD 86 d	C6 n			
ADC A, ×	8F	88	89	8A	8B	8C	8D	8E			DD 8E d	FD 8E d	CE n			
SUB ×	97	90	91	92	93	94	95	96			DD 96 d	FD 96 d	D6 n			
SBC A, ×	9F	98	99	9A	9B	9C	9D	9E			DD 9E d	FD 9E d	DE n			
AND ×	A7	A0	A1	A2	A3	A4	A5	A6			DD A6 d	FD A6 d	E6 n			
XOR ×	AF	A8	A9	AA	AB	AC	AD	AE			DD AE d	FD AE d	EE n			
OR ×	B7	B0	B1	B2	B3	B4	B5	B6			DD B6 d	FD B6 d	F6 n			
CP ×	BF	B8	B9	BA	BB	BC	BD	BE			DD BE d	FD BE d	FE n			
INC ×	3C	04	0C	14	1C	24	2C	34			DD 34 d	FD 34 d				
DEC ×	3D	05	0D	15	1D	25	2D	35			DD 35 d	FD 35 d				

《付録1-ⓑ》Z-80活用表

回転

×	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
RLC ×	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
RRC ×	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
RL ×	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
RR ×	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
SLA ×	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
SRA ×	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
SRL ×	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
RLD								ED 6F		
RRD								ED 67		

RLCA	07
RRCA	0F
RLA	17
RRA	1F

16ビット

×	BC	DE	HL	SP	IX	IY	AF	nn	(nn)
LD AF, ×									
LD BC, ×								01 n n	ED 4B n n
LD DE, ×								11 n n	ED 5B n n
LD HL, ×								21 n n	2A n n
LD SP, ×			F9		DD F9	FD F9		31 n n	ED 7B n n
LD IX, ×								DD 21 n n	DD 2A n n
LD IY, ×								FD 21 n n	FD 2A n n
LD (nn), ×	ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
PUSH ×	C5	D5	E5		DD E5	FD E5	F5		
POP ×	C1	D1	E1		DD E1	FD E1	F1		
ADD HL, ×	09	19	29	39					
ADD IX, ×	DD 09	DD 19		DD 39	DD 29				
ADD IY, ×	FD 09	FD 19		FD 39		FD 29			
ADC HL, ×	ED 4A	ED 5A	ED 6A	ED 7A					
SBC HL, ×	ED 42	ED 52	ED 62	ED 72					
INC ×	03	13	23	33	DD 23	FD 23			
DEC ×	0B	1B	2B	3B	DD 2B	FD 2B			

《付録1-ⓒ》Z-80活用表

ジャンプ，コール，リターン

×	UN COND	C	NC	Z	NZ	PE	PO	M	P	
JP ×, nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JR ×, e	18 e−2	38 e−2	30 e−2	28 e−2	20 e−2					
JP (HL)	E9									
JP (IX)	DD E9									
JP (IY)	FD E9									
CALL ×, nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
DJNZ e										10 e−2
RET ×	C9	D8	D0	C8	C0	E8	E0	F8	F0	
RETI	ED 4D									
RETN	ED 45									

入　力

IN A, n	DB n
IN A, (C)	ED 78
IN B, (C)	ED 40
IN C, (C)	ED 48
IN D, (C)	ED 50
IN E, (C)	ED 58
IN H, (C)	ED 60
IN L, (C)	ED 68
INI	ED A2
INIR	ED B2
IND	ED AA
INDR	ED BA

ブロック・サーチ

CPI	ED A1
CPIR	ED B1
CPD	ED A9
CPDR	ED B9

ブロック転送

LDI	ED A0
LDIR	ED B0
LDD	ED A8
LDDR	ED B8

アキュムレータ操作

DAA	27
CPL	2F
NEG	ED 44
CCF	3F
SCF	37

エクスチェンジ

EX AF, AF'	08
EX DE, HL	EB
EX (SP), HL	E3
EX (SP), IX	DD E3
EX (SP), IY	FD E3
EXX	D9

リスタート

RST 00H	C7
RST 08H	CF
RST 10H	D7
RST 18H	DF
RST 20H	E7
RST 28H	EF
RST 30H	F7
RST 38H	FF

CPUコントロール

NOP	00
HALT	76
DI	F3
EI	FB
IM 0	ED 46
IM 1	ED 56
IM 2	ED 5E

出　力

OUT n, A	D3 n
OUT (C), A	ED 79
OUT (C), B	ED 41
OUT (C), C	ED 49
OUT (C), D	ED 51
OUT (C), E	ED 59
OUT (C), H	ED 61
OUT (C), L	ED 69
OUTI	ED A3
OTIR	ED B3
OUTD	ED AB
OTDR	ED BB

《付録1-ⓓ》Z-80活用法

ビット操作

＼　×	A	B	C	D	E	H	L	(HL)	(I× +d)	(IY +d)
BIT 0, ×	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
BIT 1, ×	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
BIT 2, ×	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
BIT 3, ×	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
BIT 4, ×	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
BIT 5, ×	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
BIT 6, ×	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76
BIT 7, ×	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
RES 0, ×	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
RES 1, ×	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
RES 2, ×	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
RES 3, ×	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
RES 4, ×	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
RES 5, ×	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
RES 6, ×	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
RES 7, ×	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
SET 0, ×	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
SET 1, ×	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
SET 2, ×	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
SET 3, ×	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
SET 4, ×	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
SET 5, ×	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
SET 6, ×	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
SET 7, ×	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

《付録2-ⓐ》機械語↔ニーモニック対応表

機械語 ←→ ニーモニック

機械語	ニーモニック		機械語	ニーモニック		機械語	ニーモニック		機械語	ニーモニック	
00	NOP		40	LD	B,B	80	ADD	A,B	C0	RET	NZ
01	LD	BC,nn	41	LD	B,C	81	ADD	A,C	C1	POP	BC
02	LD	(BC),A	42	LD	B,D	82	ADD	A,D	C2	JP	NZ,nn
03	INC	BC	43	LD	B,E	83	ADD	A,E	C3	JP	nn
04	INC	B	44	LD	B,H	84	ADD	A,H	C4	CALL	NZ,nn
05	DEC	B	45	LD	B,L	85	ADD	A,L	C5	PUSH	BC
06	LD	B,n	46	LD	B,(HL)	86	ADD	A,(HL)	C6	ADD	A,n
07	RLCA		47	LD	B,A	87	ADD	A,A	C7	RST	00H
08	EX	AF,AF'	48	LD	C,B	88	ADC	A,B	C8	RET	Z
09	ADD	HL,BC	49	LD	C,C	89	ADC	A,C	C9	RET	
0A	LD	A,(BC)	4A	LD	C,D	8A	ADC	A,D	CA	JP	Z,nn
0B	DEC	BC	4B	LD	C,E	8B	ADC	A,E	CB		
0C	INC	C	4C	LD	C,H	8C	ADC	A,H	CC	CALL	Z,nn
0D	DEC	C	4D	LD	C,L	8D	ADC	A,L	CD	CALL	nn
0E	LD	C,n	4E	LD	C,(HL)	8E	ADC	A,(HL)	CE	ADC	A,n
0F	RRCA		4F	LD	C,A	8F	ADC	A,A	CF	RST	08H
10	DJNZ	e	50	LD	D,B	90	SUB	B	D0	RET	NC
11	LD	DE,nn	51	LD	D,C	91	SUB	C	D1	POP	DE
12	LD	(DE),A	52	LD	D,D	92	SUB	D	D2	JP	NC,nn
13	INC	DE	53	LD	D,E	93	SUB	E	D3	OUT	n,A
14	INC	D	54	LD	D,H	94	SUB	H	D4	CALL	NC,nn
15	DEC	D	55	LD	D,L	95	SUB	L	D5	PUSH	DE
16	LD	D,n	56	LD	D,(HL)	96	SUB	(HL)	D6	SUB	n
17	RLA		57	LD	D,A	97	SUB	A	D7	RST	10H
18	JR	e	58	LD	E,B	98	SBC	A,B	D8	RET	C
19	ADD	HL,DE	59	LD	E,C	99	SBC	A,C	D9	EXX	
1A	LD	A,(DE)	5A	LD	E,D	9A	SBC	A,D	DA	JP	C,nn
1B	DEC	DE	5B	LD	E,E	9B	SBC	A,E	DB	IN	A,n
1C	INC	E	5C	LD	E,H	9C	SBC	A,H	DC	CALL	C,nn
1D	DEC	E	5D	LD	E,L	9D	SBC	A,L	DD		
1E	LD	E,n	5E	LD	E,(HL)	9E	SBC	A,(HL)	DE	SBC	A,n
1F	RRA		5F	LD	E,A	9F	SBC	A,A	DF	RST	18H
20	JR	NZ,e	60	LD	H,B	A0	AND	B	E0	RET	PO
21	LD	HL,nn	61	LD	H,C	A1	AND	C	E1	POP	HL
22	LD	(nn),HL	62	LD	H,D	A2	AND	D	E2	JP	PO,nn
23	INC	HL	63	LD	H,E	A3	AND	E	E3	EX	(SP),HL
24	INC	H	64	LD	H,H	A4	AND	H	E4	CALL	PO,nn
25	DEC	H	65	LD	H,L	A5	AND	L	E5	PUSH	HL
26	LD	H,n	66	LD	H,(HL)	A6	AND	(HL)	E6	AND	n
27	DAA		67	LD	H,A	A7	AND	A	E7	RST	20H
28	JR	Z,e	68	LD	L,B	A8	XOR	B	E8	RET	PE
29	ADD	HL,HL	69	LD	L,C	A9	XOR	C	E9	JP	(HL)
2A	LD	HL,(nn)	6A	LD	L,D	AA	XOR	D	EA	JP	PE,nn
2B	DEC	HL	6B	LD	L,E	AB	XOR	E	EB	EX	DE,HL
2C	INC	L	6C	LD	L,H	AC	XOR	H	EC	CALL	PE,nn
2D	DEC	L	6D	LD	L,L	AD	XOR	L	ED		
2E	LD	L,n	6E	LD	L,(HL)	AE	XOR	(HL)	EE	XOR	n
2F	CPL		6F	LD	L,A	AF	XOR	A	EF	RST	28H
30	JR	NC,e	70	LD	(HL),B	B0	OR	B	F0	RET	P
31	LD	SP,nn	71	LD	(HL),C	B1	OR	C	F1	POP	AF
32	LD	(nn),A	72	LD	(HL),D	B2	OR	D	F2	JP	P,nn
33	INC	SP	73	LD	(HL),E	B3	OR	E	F3	DI	
34	INC	(HL)	74	LD	(HL),H	B4	OR	H	F4	CALL	P,nn
35	DEC	(HL)	75	LD	(HL),L	B5	OR	L	F5	PUSH	AF
36	LD	(HL),n	76	HALT		B6	OR	(HL)	F6	OR	n
37	SCF		77	LD	(HL),A	B7	OR	A	F7	RST	30H
38	JR	C,e	78	LD	A,B	B8	CP	B	F8	RET	M
39	ADD	HL,SP	79	LD	A,C	B9	CP	C	F9	LD	SP,HL
3A	LD	A,(nn)	7A	LD	A,D	BA	CP	D	FA	JP	M,nn
3B	DEC	SP	7B	LD	A,E	BB	CP	E	FB	EI	
3C	INC	A	7C	LD	A,H	BC	CP	H	FC	CALL	M,nn
3D	DEC	A	7D	LD	A,L	BD	CP	L	FD		
3E	LD	A,n	7E	LD	A,(HL)	BE	CP	(HL)	FE	CP	n
3F	CCF		7F	LD	A,A	BF	CP	A	FF	RST	38H

《付録2-ⓑ》機械語↔ニーモニック対応表

CB ××

コード	命令	オペランド	コード	命令	オペランド	コード	命令	オペランド	コード	命令	オペランド
00	RLC	B	40	BIT	0,B	80	RES	0,B	C0	SET	0,B
01	RLC	C	41	BIT	0,C	81	RES	0,C	C1	SET	0,C
02	RLC	D	42	BIT	0,D	82	RES	0,D	C2	SET	0,D
03	RLC	E	43	BIT	0,E	83	RES	0,E	C3	SET	0,E
04	RLC	H	44	BIT	0,H	84	RES	0,H	C4	SET	0,H
05	RLC	L	45	BIT	0,L	85	RES	0,L	C5	SET	0,L
06	RLC	(HL)	46	BIT	0,(HL)	86	RES	0,(HL)	C6	SET	0,(HL)
07	RLC	A	47	BIT	0,A	87	RES	0,A	C7	SET	0,A
08	RRC	B	48	BIT	1,B	88	RES	1,B	C8	SET	1,B
09	RRC	C	49	BIT	1,C	89	RES	1,C	C9	SET	1,C
0A	RRC	D	4A	BIT	1,D	8A	RES	1,D	CA	SET	1,D
0B	RRC	E	4B	BIT	1,E	8B	RES	1,E	CB	SET	1,E
0C	RRC	H	4C	BIT	1,H	8C	RES	1,H	CC	SET	1,H
0D	RRC	L	4D	BIT	1,L	8D	RES	1,L	CD	SET	1,L
0E	RRC	(HL)	4E	BIT	1,(HL)	8E	RES	1,(HL)	CE	SET	1,(HL)
0F	RRC	A	4F	BIT	1,A	8F	RES	1,A	CF	SET	1,A
10	RL	B	50	BIT	2,B	90	RES	2,B	D0	SET	2,B
11	RL	C	51	BIT	2,C	91	RES	2,C	D1	SET	2,C
12	RL	D	52	BIT	2,D	92	RES	2,D	D2	SET	2,D
13	RL	E	53	BIT	2,E	93	RES	2,E	D3	SET	2,E
14	RL	H	54	BIT	2,H	94	RES	2,H	D4	SET	2,H
15	RL	L	55	BIT	2,L	95	RES	2,L	D5	SET	2,L
16	RL	(HL)	56	BIT	2,(HL)	96	RES	2,(HL)	D6	SET	2,(HL)
17	RL	A	57	BIT	2,A	97	RES	2,A	D7	SET	2,A
18	RR	B	58	BIT	3,B	98	RES	3,B	D8	SET	3,B
19	RR	C	59	BIT	3,C	99	RES	3,C	D9	SET	3,C
1A	RR	D	5A	BIT	3,D	9A	RES	3,D	DA	SET	3,D
1B	RR	E	5B	BIT	3,E	9B	RES	3,E	DB	SET	3,E
1C	RR	H	5C	BIT	3,H	9C	RES	3,H	DC	SET	3,H
1D	RR	L	5D	BIT	3,L	9D	RES	3,L	DD	SET	3,L
1E	RR	(HL)	5E	BIT	3,(HL)	9E	RES	3,(HL)	DE	SET	3,(HL)
1F	RR	A	5F	BIT	3,A	9F	RES	3,A	DF	SET	3,A
20	SLA	B	60	BIT	4,B	A0	RES	4,B	E0	SET	4,B
21	SLA	C	61	BIT	4,C	A1	RES	4,C	E1	SET	4,C
22	SLA	D	62	BIT	4,D	A2	RES	4,D	E2	SET	4,D
23	SLA	E	63	BIT	4,E	A3	RES	4,E	E3	SET	4,E
24	SLA	H	64	BIT	4,H	A4	RES	4,H	E4	SET	4,H
25	SLA	L	65	BIT	4,L	A5	RES	4,L	E5	SET	4,L
26	SLA	(HL)	66	BIT	4,(HL)	A6	RES	4,(HL)	E6	SET	4,(HL)
27	SLA	A	67	BIT	4,A	A7	RES	4,A	E7	SET	4,A
28	SRA	B	68	BIT	5,B	A8	RES	5,B	E8	SET	5,B
29	SRA	C	69	BIT	5,C	A9	RES	5,C	E9	SET	5,C
2A	SRA	D	6A	BIT	5,D	AA	RES	5,D	EA	SET	5,D
2B	SRA	E	6B	BIT	5,E	AB	RES	5,E	EB	SET	5,E
2C	SRA	H	6C	BIT	5,H	AC	RES	5,H	EC	SET	5,H
2D	SRA	L	6D	BIT	5,L	AD	RES	5,L	ED	SET	5,L
2E	SRA	(HL)	6E	BIT	5,(HL)	AE	RES	5,(HL)	EE	SET	5,(HL)
2F	SRA	A	6F	BIT	5,A	AF	RES	5,A	EF	SET	5,A
30			70	BIT	6,B	B0	RES	6,B	F0	SET	6,B
31			71	BIT	6,C	B1	RES	6,C	F1	SET	6,C
32			72	BIT	6,D	B2	RES	6,D	F2	SET	6,D
33			73	BIT	6,E	B3	RES	6,E	F3	SET	6,E
34			74	BIT	6,H	B4	RES	6,H	F4	SET	6,H
35			75	BIT	6,L	B5	RES	6,L	F5	SET	6,L
36			76	BIT	6,(HL)	B6	RES	6,(HL)	F6	SET	6,(HL)
37			77	BIT	6,A	B7	RES	6,A	F7	SET	6,A
38	SRL	B	78	BIT	7,B	B8	RES	7,B	F8	SET	7,B
39	SRL	C	79	BIT	7,C	B9	RES	7,C	F9	SET	7,C
3A	SRL	D	7A	BIT	7,D	BA	RES	7,D	FA	SET	7,D
3B	SRL	E	7B	BIT	7,E	BB	RES	7,E	FB	SET	7,E
3C	SRL	H	7C	BIT	7,H	BC	RES	7,H	FC	SET	7,H
3D	SRL	L	7D	BIT	7,L	BD	RES	7,L	FD	SET	7,L
3E	SRL	(HL)	7E	BIT	7,(HL)	BE	RES	7,(HL)	FE	SET	7,(HL)
3F	SRL	A	7F	BIT	7,A	BF	RES	7,A	FF	SET	7,A

《付録2-ⓒ》機械語↔ニーモニック対応表

DD ××		
09	ADD	IX, BC
19	ADD	IX, DE
21	LD	IX, nn
22	LD	(nn), IX
23	INC	IX
29	ADD	IX, IX
2A	LD	IX, (nn)
2B	DEC	IX
34	INC	(IX+d)
35	DEC	(IX+d)
36	LD	(IX+d), n
39	ADD	IX, SP
46	LD	B, (IX+d)
4E	LD	C, (IX+d)
56	LD	D, (IX+d)
5E	LD	E, (IX+d)
66	LD	H, (IX+d)
6E	LD	L, (IX+d)
70	LD	(IX+d), B
71	LD	(IX+d), C
72	LD	(IX+d), D
73	LD	(IX+d), E
74	LD	(IX+d), H
75	LD	(IX+d), L
77	LD	(IX+d), A
7E	LD	A, (IX+d)
86	ADD	A, (IX+d)
8E	ADC	A, (IX+d)
96	SUB	(IX+d)
9E	SBC	A, (IX+d)
A6	AND	(IX+d)
AE	XOR	(IX+d)
B6	OR	(IX+d)
BE	CP	(IX+d)
CB d 06	RLC	(IX+d)
CB d 0E	RRC	(IX+d)
CB d 16	RL	(IX+d)
CB d 1E	RR	(IX+d)
CB d 26	SLA	(IX+d)
CB d 2E	SRA	(IX+d)
CB d 3E	SRL	(IX+d)
CB d 46	BIT	0, (IX+d)
CB d 4E	BIT	1, (IX+d)
CB d 56	BIT	2, (IX+d)
CB d 5E	BIT	3, (IX+d)
CB d 66	BIT	4, (IX+d)
CB d 6E	BIT	5, (IX+d)
CB d 76	BIT	6, (IX+d)
CB d 7E	BIT	7, (IX+d)
CB d 86	RES	0, (IX+d)
CB d 8E	RES	1, (IX+d)
CB d 96	RES	2, (IX+d)
CB d 9E	RES	3, (IX+d)
CB d A6	RES	4, (IX+d)
CB d AE	RES	5, (IX+d)
CB d B6	RES	6, (IX+d)
CB d BE	RES	7, (IX+d)
CB d C6	SET	0, (IX+d)
CB d CE	SET	1, (IX+d)
CB d D6	SET	2, (IX+d)
CB d DE	SET	3, (IX+d)
CB d E6	SET	4, (IX+d)
CB d EE	SET	5, (IX+d)
CB d F6	SET	6, (IX+d)
CB d FE	SET	7, (IX+d)
E1	POP	IX
E3	EX	(SP), IX
E5	PUSH	IX
E9	JP	(IX)
F9	LD	SP, IX

ED ××		
40	IN	B, (C)
41	OUT	(C), B
42	SBC	HL, BC
43	LD	(nn), BC
44	NEG	
45	RETN	
46	IM	0
47	LD	I, A
48	IN	C, (C)
49	OUT	(C), C
4A	ADC	HL, BC
4B	LD	BC, (nn)
4D	RETI	
4F	LD	R, A
50	IN	D, (C)
51	OUT	(C), D
52	SBC	HL, DE
53	LD	(nn), DE
56	IM	1
57	LD	A, I
58	IN	E, (C)
59	OUT	(C), E
5A	ADC	HL, DE
5B	LD	DE, (nn)
5E	IM	2
5F	LD	A, R
60	IN	H, (C)
61	OUT	(C), H
62	SBC	HL, HL
67	RRD	
68	IN	L, (C)
69	OUT	(C), L
6A	ADC	HL, HL
6F	RLD	
72	SBC	HL, SP
73	LD	(nn), SP
78	IN	A, (C)
79	OUT	(C), A
7A	ADC	HL, SP
7B	LD	SP, (nn)
A0	LDI	
A1	CPI	
A2	INI	
A3	OUTI	
A8	LDD	
A9	CPD	
AA	IND	
AB	OUTD	
B0	LDIR	
B1	CPIR	
B2	INIR	
B3	OTIR	
B8	LDDR	
B9	CPDR	
BA	INDR	
BB	OTDR	

FD ××		
09	ADD	IY, BC
19	ADD	IY, DE
21	LD	IY, nn
22	LD	(nn), IY
23	INC	IY
29	ADD	IY, IY
2A	LD	IY, (nn)
2B	DEC	IY
34	INC	(IY+d)
35	DEC	(IY+d)
36	LD	(IY+d), n
39	ADD	IY, SP
46	LD	B, (IY+d)
4E	LD	C, (IY+d)
56	LD	D, (IY+d)
5E	LD	E, (IY+d)
66	LD	H, (IY+d)
6E	LD	L, (IY+d)
70	LD	(IY+d), B
71	LD	(IY+d), C
72	LD	(IY+d), D
73	LD	(IY+d), E
74	LD	(IY+d), H
75	LD	(IY+d), L
77	LD	(IY+d), A
7E	LD	A, (IY+d)
86	ADD	A, (IY+d)
8E	ADC	A, (IY+d)
96	SUB	(IY+d)
9E	SBC	A, (IY+d)
A6	AND	(IY+d)
AE	XOR	(IY+d)
B6	OR	(IY+d)
BE	CP	(IY+d)
CB d 06	RLC	(IY+d)
CB d 0E	RRC	(IY+d)
CB d 16	RL	(IY+d)
CB d 1E	RR	(IY+d)
CB d 26	SLA	(IY+d)
CB d 2E	SRA	(IY+d)
CB d 3E	SRL	(IY+d)
CB d 46	BIT	0, (IY+d)
CB d 4E	BIT	1, (IY+d)
CB d 56	BIT	2, (IY+d)
CB d 5E	BIT	3, (IY+d)
CB d 66	BIT	4, (IY+d)
CB d 6E	BIT	5, (IY+d)
CB d 76	BIT	6, (IY+d)
CB d 7E	BIT	7, (IY+d)
CB d 86	RES	0, (IY+d)
CB d 8E	RES	1, (IY+d)
CB d 96	RES	2, (IY+d)
CB d 9E	RES	3, (IY+d)
CB d A6	RES	4, (IY+d)
CB d AE	RES	5, (IY+d)
CB d B6	RES	6, (IY+d)
CB d BE	RES	7, (IY+d)
CB d C6	SET	0, (IY+d)
CB d CE	SET	1, (IY+d)
CB d D6	SET	2, (IY+d)
CB d DE	SET	3, (IY+d)
CB d E6	SET	4, (IY+d)
CB d EE	SET	5, (IY+d)
CB d F6	SET	6, (IY+d)
CB d FE	SET	7, (IY+d)
E1	POP	IY
E3	EX	(SP), IY
E5	PUSH	IY
E9	JP	(IY)
F9	LD	SP, IY

《付録3》PC-8001キャラクタ・コード表

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		D_E		0	@	P		p	▁	┴		ー	タ	ミ	═	╳
1	S_H	D_1	!	1	A	Q	a	q	▂	┬	。	ア	チ	ム	╞	円
2	S_X	D_2	‖	2	B	R	b	r	▃	┤	「	イ	ツ	メ	╪	年
3	E_X	D_3	#	3	C	S	c	s	▄	├	」	ウ	テ	モ	╡	月
4	E_T	D_4・	$	4	D	T	d	t	▅	▔	、	エ	ト	ヤ	◢	日
5	E_Q	N_K	%	5	E	U	e	u	▆	─	・	オ	ナ	ユ	◣	時
6	A_K	S_N	&	6	F	V	f	v	▇	│	ヲ	カ	ニ	ヨ	◥	分
7	B_L	E_B	▼	7	G	W	g	w	█	▕	ァ	キ	ヌ	ラ	◤	秒
8	B_S	C_N	(	8	H	X	h	x	▏	┌	ィ	ク	ネ	リ	♠	
9	H_T	E_M	)	9	I	Y	i	y	▎	┐	ゥ	ケ	ノ	ル	♥	
A	L_F	S_B	*	:	J	Z	j	z	▍	└	ェ	コ	ハ	レ	♦	
B	H_M	E_C	+	;	K	[	k	{	▌	┘	ォ	サ	ヒ	ロ	♣	
C	C_L	→	,	<	L	¥	l	¦	▋	╭	ャ	シ	フ	ワ	●	
D	C_R	←	−	=	M	]	m	}	▊	╮	ュ	ス	ヘ	ン	○	
E	S_O	↑	.	>	N	^	n	~	▉	╰	ョ	セ	ホ	゛	╱	
F	S_I	↓	/	?	O	_	o		┼	╯	ッ	ソ	マ	゜	╲	

《付録4》10進↔16進変換表

下位 / 上位	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

最上位× / 桁	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
×00	256	512	768	1024	1280	1536	1792	2048	2304	2560	2816	3072	3328	3584	3840
×000	4096	8192	12288	16384	20480	24576	28672	32768	36864	40960	45056	49152	53248	57344	61440
×0000	65536														

《付録5》2 進↔16進変換表

16 進 数	2 進 数
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
A	1 0 1 0
B	1 0 1 1
C	1 1 0 0
D	1 1 0 1
E	1 1 1 0
F	1 1 1 1

《付録6》レジスタの種類

種　類	記号	名　　称	ビット
専用レジスタ	PC	プログラム・カウンタ	16
	SP	スタック・ポインタ	16
	IX	インデックス・レジスタ	16
	IY	インデックス・レジスタ	16
	I	インタラプト・ページ・アドレス・レジスタ	8
	R	メモリ・リフレッシュ・レジスタ	8
アキュームレータとフラグ	A	アキューム・レータ	8
	F	フラグ	8
	A′	アキューム・レータ(サブ)	8
	F′	フラグ(サブ)	8
汎用レジスタ	B	メイン・レジスタ	8
	C		8
	D		8
	E		8
	H		8
	L		8
	B′	サブ・レジスタ	8
	C′		8
	D′		8
	E′		8
	H′		8
	L′		8

《付録7-ⓐ》Z-80 CODING SHEET
ADR ESS
1
MACHINE CODE
10
LEBEL
20
OPERATION & OPERAND
30
40
COMMENT
50
60
70
80
(ⓐ、ⓑは同じものです。B4で2頁まとめてコピーし，切断してご使用下さい)

《付録7-ⓑ》Z-80　CODING SHEET

(ⓐ, ⓑは同じものです。B 4 で 2 頁まとめてコピーし，切断してご使用下さい)

《付録8》メモリアドレスマップ

《付録9》レイアウト・シート（40×25モード）

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	F3 00	02	04	06	08	0A	0C	0E	F3 10	12	14	16	18	1A	1C	1E	F3 20	22	24	26
1	F3 78	7A	7C	7E	F3 80	82	84	86	88	8A	8C	8E	F3 90	92	94	96	98	9A	9C	9E
2	F3 F0	F2	F4	F6	F8	FA	FC	FE	F4 00	02	04	06	08	0A	0C	0E	F4 10	12	14	16
3	F4 68	6A	6C	6E	F4 70	72	74	76	78	7A	7C	7E	F4 80	82	84	86	88	8A	8C	8E
4	F4 E0	E2	E4	E6	E8	EA	EC	EE	F4 F0	F2	F4	F6	F8	FA	FC	FE	F5 00	02	04	06
5	F5 58	5A	5C	5E	F5 60	62	64	66	68	6A	6C	6E	F5 70	72	74	76	78	7A	7C	7E
6	F5 D0	D2	D4	D6	D8	DA	DC	DE	F5 E0	E2	E4	E6	E8	EA	EC	EE	F5 F0	F2	F4	F6
7	F6 48	4A	4C	4E	F6 50	52	54	56	58	5A	5C	5E	F6 60	62	64	66	68	6A	6C	6E
8	F6 C0	C2	C4	C6	C8	CA	CC	CE	F6 D0	D2	D4	D6	D8	DA	DC	DE	F6 E0	E2	E4	E6
9	F7 38	3A	3C	3E	F7 40	42	44	46	48	4A	4C	4E	F7 50	52	54	56	58	5A	5C	5E
10	F7 B0	B2	B4	B6	B8	BA	BC	BE	F7 C0	C2	C4	C6	C8	CA	CC	CE	F7 D0	D2	D4	D6
11	F8 28	2A	2C	2E	F8 30	32	34	36	38	3A	3C	3E	F8 40	42	44	46	48	4A	4C	4E
12	F8 A0	A2	A4	A6	A8	AA	AC	AE	F8 B0	B2	B4	B6	B8	BA	BC	BE	F8 C0	C2	C4	C6
13	F9 18	1A	1C	1E	F9 20	22	24	26	28	2A	2C	2E	F9 30	32	34	36	38	3A	3C	3E
14	F9 90	92	94	96	98	9A	9C	9E	F9 A0	A2	A4	A6	A8	AA	AC	AE	F9 B0	B2	B4	B6
15	FA 08	0A	DC	0E	FA 10	12	14	16	18	1A	1C	1E	FA 20	22	24	26	28	2A	2C	2E
16	FA 80	82	84	86	88	8A	8C	8E	FA 90	92	94	96	98	9A	9C	9E	FA A0	A2	A4	A6
17	FA F8	FA	FC	FE	FB 00	02	04	06	08	0A	0C	0E	FB 10	12	14	16	18	1A	1C	1E
18	FB 70	72	74	76	78	7A	7C	7E	FB 80	82	84	86	88	8A	8C	8E	FB 90	92	94	96
19	FB E8	EA	EC	EE	FB F0	F2	F4	F6	F8	FA	EC	FE	FC 00	02	04	06	08	DA	0C	0E
20	FC 60	62	64	66	68	6A	6C	6E	FC 70	72	74	76	78	7A	7C	7E	FC 80	82	84	86
21	FC D8	DA	DC	DE	FC E0	E2	E4	E6	E8	EA	EC	EE	FC F0	F2	F4	F6	F8	FA	FC	FE
22	FD 50	52	54	56	58	5A	5C	5E	FD 60	62	64	66	68	6A	6C	6E	FD 70	72	74	76
23	FD C8	CA	CC	CE	FD D0	D2	D4	D6	D8	CA	CC	CE	FD E0	E2	E4	E6	E8	EA	EC	EE
24	FE 40	42	44	46	48	4A	4C	4E	FE 50	52	54	56	58	5A	5C	5E	FE 60	62	64	66

	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
0	28	2A	2C	2E	F3 30	32	34	36	38	3A	3C	3E	F3 40	42	44	46	48	4A	4C	4E
1	F3 A0	A2	A4	A6	A8	AA	AC	AE	F3 B0	B2	B4	B6	B8	BA	BC	BE	F3 C0	C2	C4	C6
2	18	1A	1C	1E	F4 20	22	24	26	28	2A	2C	2E	F4 30	32	34	36	38	3A	3C	3E
3	F4 90	92	94	96	98	9A	9C	9E	F4 A0	A2	A4	A6	A8	AA	AC	AE	F4 B0	B2	B4	B6
4	28	0A	0C	0E	F5 10	12	14	16	18	1A	1C	1E	F5 20	22	24	26	28	2A	2C	2E
5	F5 08	82	84	86	88	8A	8C	8E	F5 90	92	94	96	98	9A	9C	9E	F5 A0	A2	A4	A6
6	F8	FA	FC	FE	F6 00	02	04	06	08	0A	0C	0E	F6 10	12	14	16	18	1A	1C	1E
7	F6 70	72	74	76	78	7A	7C	7E	F6 80	82	84	86	88	8A	8C	8E	F6 90	92	94	96
8	E8	EA	EC	EE	F6 F0	F2	F4	F6	F8	FA	FC	FE	F7 00	02	04	06	08	0A	0C	0E
9	F7 60	62	64	66	68	6A	6C	6E	F7 70	72	74	76	78	7A	7C	7E	F7 80	82	84	86
10	D8	DA	DC	DE	F7 E0	E2	E4	E6	E8	EA	EC	EE	F7 F0	F2	F4	F6	F8	FA	FC	FE
11	F8 50	52	54	56	58	5A	5C	5E	F8 60	62	64	66	68	6A	6C	6E	F8 70	72	74	76
12	C8	CA	CC	CE	F8 D0	D2	D4	D6	D8	DA	DC	DE	F8 E0	E2	E4	E6	E8	EA	EC	EE
13	F9 40	42	44	46	48	4A	4C	4E	F9 50	52	54	56	58	5A	5C	5E	F9 60	62	64	66
14	B8	BA	BC	BE	F9 C0	C2	C4	C6	C8	CA	CC	CE	F9 D0	D2	D4	D6	D8	DA	DC	DE
15	FA 30	32	34	36	38	3A	3C	3E	FA 40	42	44	46	48	4A	4C	4E	FA 50	52	54	56
16	A8	AA	AC	AE	FA B0	B2	B4	B6	B8	BA	BC	BE	FA C0	C2	C4	C6	C8	CA	CC	CE
17	FB 20	22	24	26	28	2A	2C	2E	FB 30	32	34	36	38	3A	3C	3E	FB 40	42	44	46
18	98	9A	9C	9E	FB A0	A2	A4	A6	A8	AA	AC	AE	FB B0	B2	B4	B6	B8	BA	BC	BE
19	FC 10	12	14	16	18	1A	1C	1E	FC 20	22	24	26	28	2A	2C	2E	FC 30	32	34	36
20	88	8A	8C	8E	FC 90	92	94	96	98	9A	9C	9E	FC A0	A2	A4	A6	A8	AA	AC	AE
21	FD 00	02	04	06	08	0A	0C	0E	FD 10	12	14	16	18	1A	1C	1E	FD 20	22	24	26
22	78	7A	7C	7E	FD 80	82	84	86	88	8A	8C	8E	FD 90	92	94	96	98	9A	9C	9E
23	FD F0	F2	F4	F6	F8	FA	FC	FE	FE 00	02	04	06	08	0A	0C	0E	FE 10	12	14	16
24	68	6A	6C	6E	FE 70	72	74	76	78	7A	7C	7E	FE 80	82	84	86	88	8A	8C	8E

《付録10》レイアウト・シート(80×25モード)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
0	F3 00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	F3 10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	F3 20	21	22	23	24	25	26	27
1	F3 78	79	7A	7B	7C	7D	7E	7F	F3 80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	F3 90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
2	F3 F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	F4 00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	F4 10	11	12	13	14	15	16	17
3	F4 68	69	6A	6B	6C	6D	6E	6F	F4 70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	F4 80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
4	F4 E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F4 F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	F5 00	01	02	03	04	05	06	07
5	F5 58	59	5A	5B	5C	5D	5E	5F	F5 60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	F5 70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
6	F5 D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	F5 E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F5 F0	F1	F2	F3	F4	F5	F6	F7
7	F6 48	49	4A	4B	4C	4D	4E	4F	F6 50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	F6 60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
8	F6 C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	F6 D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	F6 E0	E1	E2	E3	E4	E5	E6	E7
9	F7 38	39	3A	3B	3C	3D	3E	3F	F7 40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	F7 50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
10	F7 B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	F7 C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	F7 D0	D1	D2	D3	D4	D5	D6	D7
11	F8 28	29	2A	2B	2C	2D	2E	2F	F8 30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	F8 40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
12	F8 A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	F8 B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	F8 C0	C1	C2	C3	C4	C5	C6	C7
13	F9 18	19	1A	1B	1C	1D	1E	1F	F9 20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	F9 30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
14	F9 90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	F9 A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	F9 B0	B1	B2	B3	B4	B5	B6	B7
15	FA 08	09	0A	0B	0C	0D	0E	0F	FA 10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	FA 20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
16	FA 80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	FA 90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	FA A0	A1	A2	A3	A4	A5	A6	A7
17	FA F8	F9	FA	FB	FC	FD	FE	FF	FB 00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	FB 10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
18	FB 70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	FB 80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	FB 90	91	92	93	94	95	96	97
19	FB E8	E9	EA	EB	EC	ED	EE	EF	FB F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	FC 00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
20	FC 60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	FC 70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	FC 80	81	82	83	84	85	86	87
21	FC D8	D9	DA	DB	DC	DD	DE	DF	FC E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	FC F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF
22	FD 50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	FD 60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	FD 70	71	72	73	74	75	76	77
23	FD C8	C9	CA	CB	CC	CD	CE	CF	FD D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	FD E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
24	FE 40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	FE 50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	FE 60	61	62	63	64	65	66	67

	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
0	28	29	2A	2B	2C	2D	2E	2F	F3 30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	F3 40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
1	F3 A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	F3 B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	F3 C0	C1	C2	C3	C4	C5	C6	C7
2	18	19	1A	1B	1C	1D	1E	1F	F4 20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	F4 30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
3	F4 90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	F4 A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	F4 B0	B1	B2	B3	B4	B5	B6	B7
4	08	09	0A	0B	0C	0D	0E	0F	F5 10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	F5 20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
5	F5 80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	F5 90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	F5 A0	A1	A2	A3	A4	A5	A6	A7
6	F8	F9	FA	FB	FC	FD	FE	FF	F6 00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	F6 10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
7	F6 70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	F6 80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	F6 90	91	92	93	94	95	96	97
8	E8	E9	EA	EB	EC	ED	EE	EF	F6 F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	F7 00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
9	F7 60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	F7 70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	F7 80	81	82	83	84	85	86	87
10	D8	D9	DA	DB	DC	DD	DE	DF	F7 E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F7 F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF
11	F8 50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	F8 60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	F8 70	71	72	73	74	75	76	77
12	C8	C9	CA	CB	CC	CD	CE	CF	F8 D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	F8 E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
13	F9 40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	F9 50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	F9 60	61	62	63	64	65	66	67
14	B8	B9	BA	BB	BC	BD	BE	BF	F9 C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	F9 D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
15	FA 30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	FA 40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	FA 50	51	52	53	54	55	56	57
16	A8	A9	AA	AB	AC	AD	AE	AF	FA B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	FA C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
17	FB 20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	FB 30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	FB 40	41	42	43	44	45	46	47
18	98	99	9A	9B	9C	9D	9E	9F	FB A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	FB B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
19	FC 10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	FC 20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	FC 30	31	32	33	34	35	36	37
20	88	89	8A	8B	8C	8D	8E	8F	FC 90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	FC A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
21	FD 00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	FD 10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	FD 20	21	22	23	24	25	26	27
22	78	79	7A	7B	7C	7D	7E	7F	ED 80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	FD 90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
23	FD F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	FE 00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	FE 10	11	12	13	14	15	16	17
24	68	69	6A	6B	6C	6D	6E	6F	FE 70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	FE 80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F

《付録11》1バイト符号付16進数

上位＼下位	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	−128	−127	−126	−125	−124	−123	−122	−121	−120	−119	−118	−117	−116	−115	−114	−113
9	−112	−111	−110	−109	−108	−107	−106	−105	−104	−103	−102	−101	−100	−99	−98	−97
A	−96	−95	−94	−93	−92	−91	−90	−89	−88	−87	−86	−85	−84	−83	−82	−81
B	−80	−79	−78	−77	−76	−75	−74	−73	−72	−71	−70	−69	−68	−67	−66	−65
C	−64	−63	−62	−61	−60	−59	−58	−57	−56	−55	−54	−53	−52	−51	−50	−49
D	−48	−47	−46	−45	−44	−43	−42	−41	−40	−39	−38	−37	−36	−35	−34	−33
E	−32	−31	−30	−29	−28	−27	−26	−25	−24	−23	−22	−21	−20	−19	−18	−17
F	−16	−15	−14	−13	−12	−11	−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

《付録12》命令のフラグへの影響

インストラクション	D7 S	Z		H		P/V	N	D0 C	コメント
ADD A,s;ADC A,s	↕	↕	×	↕	×	V	0	↕	8-bit add or add with carry
SUBs;SBC A,s;CPs ;NEG	↕	↕	×	↕	×	V	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator
ANDs	↕	↕	×	1	×	P	0	0	Logical operations
ORs;XORs	↕	↕	×	0	×	P	0	0	
INCs	↕	↕	×	↕	×	V	0	•	8-bit increment
DECs	↕	↕	×	↕	×	V	1	•	8-bit decrement
ADD DD,ss	•	•	×	×	×	•	0	↕	16-bit add
ADC HL,ss	↕	↕	×	×	×	V	0	↕	16-bit add with carry
SBC HL,ss	↕	↕	×	×	×	V	1	↕	16-bit subtract with carry
RLA;RLCA;RRA;RRCA	•	•	×	0	×	•	0	↕	Rotate accumulator
RLs;RLCs;RRs;RRCs; SLAs;SRAs;SRLs	↕	↕	×	0	×	P	0	↕	Rotate and shift locations
RLD;RRD	↕	↕	×	0	×	P	0	•	Rotate digit left and right
DAA	↕	↕	×	↕	×	P	•	↕	Decimal adjust accumulator
CPL	•	•	×	1	×	•	1	•	Complement accumulator
SCF	•	•	×	0	×	•	0	1	Set carry
CCF	•	•	×	×	×	•	0	↕	Complement carry
INr,(C)	↕	↕	×	0	×	P	0	•	Input register indirect
INI;IND;OUTI;OUTD	×	↕	×	×	×	×	1	•	Block input and output
INIR;INDR;OTIR;OTDR	×	1	×	×	×	×	1	•	Z=0 if B≠0 otherwise Z=1
LDI;LDD	×	×	×	0	×	↕	0	•	Block transfer instructions
LDIR;LDDR	×	×	×	0	×	0	0	•	P/V=1 if BC≠0, otherwise P/V=0
CPI;CPIR;CPD;CPDR	×	↕	×	×	×	↕	1	•	Block search instructions Z=1 if A=(HL), otherwise Z=0 P/V=1 if BC≠0, otherwise P/V=0
LD A,I;LD A,R	↕	↕	×	0	×	IFF	0	•	The content of the interrupt enable flip-flop(IFF) is copied into the P/V flag
BIT b,s	×	↕	×	1	×	×	0	•	The state of bit b of locations is copied into the Z flag

出所：付録12～付録24は日本電気株の好意により同社発行の「μCOM-82ユーザーズ・マニュアル」より転載しました。

DEMPAマイコンソフト全国取

北海道・東　北

店名	住所	電話
メディア旭川	旭川市2条21丁目	0166-32-6930
ブックス平和 本店	旭川市2条7丁目 マツカツビル	0166-23-6211
富貴堂	旭川市3条7丁目 オクノデパート地下	0166-25-3535
室蘭オーディオハムセンター	室蘭市寿町3-2-9	0143-44-6331
そうご電器 小樽店	小樽市稲穂2-5-8	0134-24-0381
〃 苫小牧店	苫小牧市王子町3-1-14	0144-35-2151
〃 旭川店	旭川市4条8丁目 7	
〃 サンデパート店	札幌市中央区南2条3丁目 サンデパート内	011-241-2501
〃 滝川店	滝川市栄町3-3-11	0125-22-2355
藤丸	滝川市栄町2-3	0125-22-2403
ＣＱジャパン	旭川市大雪通り4丁目	0166-26-7388
㈲堀川	岩内郡岩内町大和19-4	01356-2-0614
マイコンセンター 函館店	函館市美原町 イトーヨーカドー内	0138-51-8717
大阪屋	札幌市中央区北1条西3丁目	011-221-0181
コンピュータランド北海道	札幌市中央区北3条西2丁目 カミヤマビル	011-222-1088
光洋無線電機 中央店	札幌市中央区南2条西1丁目 北専ビル	011-271-4220
宝文堂	札幌市中央区南5条西3丁目11番地	011-511-6631
匡小川札幌店	札幌市中央区北5条西15丁目	011-642-4111
旭屋 札幌店	札幌市中央区南3条西4丁目 エイトビル	011-241-3007
紀伊国屋 札幌店	札幌市中央区南1条西1丁目14の2	011-231-2131
㈱電巧堂 新町プラザ店	青森市新町2-6-26	0177-23-2356
青森電子サービス	青森市造道沢64-7	0177-43-6175
ホビーショップメカーノ	岩手県花巻市若葉町798	0198-22-4183
ハムショップＴＣＳ	仙台市五橋2-5-6	0222-66-7681
マイコンショップコマツ	仙台市中央通2-2-21	0222-25-5290
仙台バイトショップ	仙台市堤通雨宮町3-18	0222-33-0256
ヒロセマイコンプラザ	仙台市国分町3-8-25	0222-25-3073
庄子デンキ長町本店	仙台市長町1-2-10	0222-48-1156
笹原デンキ	山形市諏訪町2-1-25	0236-22-3355
精工堂	釜石市大渡町2-5-10	0193-22-3495
イワテマイコンセンター	盛岡市夕顔瀬22-28	0196-54-3359
東高電機	盛岡市中央通1-11-20	
電巧堂チェン㈱	盛岡市中央通2丁目11-1	0196-54-2772
システムイン福島	福島市栄町2-29 九常ビル1F	0425-22-2621
若松ラジオセンター	会津若松市七日町1-17	0242-26-2711
いわきマイコンショップ	いわき市平字立町86	0426-23-0513
新栄堂書店	いわき市平字田町1	0246-22-1191
ヤマニ書房	いわき市平字2-7	0246-23-3481
モリタ	相馬市泉町15-6	02443-5-4336
郡山西武ブックセンター	郡山市駅前1-16-7	0249-34-6161
㈲ヤマト無線	郡山市中町15-27	0249-22-2262
若松電子	会津若松市門田町年貢町	0242-28-0149

東　海・北　陸

店名	住所	電話
㈱西武百貨店 静岡店	静岡市紺屋町6-7	0542-45-5151
勝木書店	福井市中央1-4-18	0766-24-0428
清明堂	富山市総輪3-2-24	0764-24-4166
うつのみや片町店	金沢市片町2-1-7	0762-21-6136
吉野電化	静岡県藤枝市田沼1-15-46	0546-35-1584
トヨムラ 静岡店	静岡市八幡1-4-36	0542-83-1331
宝塚マルサン書店	沼津市上本通町8	0559-63-0350
マルツ電波	浜松市板屋町390 0	0534-54-2366
西武百貨店浜松店マイコンショップ	浜松市鍛冶町15 西武百貨店浜松店5F	0534-55-0111
名古屋Byteショップ	名古屋市中区大須3-30-86 ラジオセンター1F	052-263-1629
マイコンショップ ISHIBASHI	碧南市碧南中央駅2F	0566-48-2323
ロッキー電子	安城市錦町2-3	05667-5-3736
フジ計器通信機	碧南市金山町5-88	0566-42-1305
マイクロキャビン四日市	四日市市三栄町2-13 タカラビル5F	0593-51-6482
理工産業	四日市市丸の城町4-20	0593-51-1651
河合無線津パーツ店	津市丸の内31-20 2F	0592-26-0111
フューチャーイン名古屋	名古屋市中区栄3-17-15 PAXビル	052-261-2555
カトー無線電気館5F	名古屋市中区栄3-32-28	052-262-6471
トヨムラ 名古屋店	名古屋市中区大須3-30-86	052-263-1660
栄電社	名古屋市中村区名駅4-23-11	052-581-1231
ほんぶん書店	岡崎市本町通2-10	0564-22-0243
丸善ブックメイツ	名古屋市中区錦3-15-13 セントラルパーク	052-971-1231
近鉄星野書店	名古屋市中村区名駅1-2-2 近鉄ビル6F	052-582-3411
イースター	名古屋市千種区東山通4-2-1	052-782-6666
三洋堂書店 いりなか店	名古屋市昭和区隼人町7-1	052-832-8202
関東BYTEショップ名古屋店	名古屋市中区大須3-30-86	052-263-1693
三洋堂書店 勝川店	春日井市勝川町7-1	0568-32-7806
〃 刈谷店	刈谷市桜町1-24	0566-24-1134
精文館	豊橋市広小路1-6	0532-54-2345
小川無線商会	大垣市宮町56-1	0584-78-6378
タケイムセン	美濃加茂市太田町2736-1	05742-6-2882
岐阜マイコンセンター	岐阜市鹿島町8-43 錦声ビル2F	0582-51-6338
フューチャーイン岐阜	岐阜市金宝町2-6 森麻ビル2F	0582-66-5911
自由書房	岐阜市神田町4-9	0582-65-4301
キトウ書店	岩倉市本町下市場125	0587-66-7070
三重通信	松阪市黒田町106	0598-23-4993
フレンド通信	福井市学園2-1-5	0776-23-3302
北都電機 北	富山市室町通り1-3-8	0764-91-1282
I・Oデータ機器	金沢市高岡市7-22	0762-21-4812
城東	小松市御宮町62	0761-24-1166
センチュリー	鯖江市東米岡2-31	0778-52-4623
ラジオタバタ	坂井郡芦原町二面4丁目605	077678 - 5525
沢田通信	礪波市千代288	07633 - 33654
パスカルイン	富山市五福10区4485	0764-41-9936
サンミュージック	金沢南局内野々市町扇ヶ丘160	0762-46-6151
橋本産業㈱富山工場	富山市木場町2-32	0764-32-7541
㈱マコム	金沢市泉野出町4-1-16	0762-43-7630
無線パーツ㈱ 金沢店	金沢市西泉2-28	0762-44-3070
電化のナカダ	東礪波郡福野町横町442-5	07632 － 4337
㈱アサヒ通商	金沢市泉本町6-36 有松ハイツ	0762-41-2822
バイナリーシステムズ	福井市春山2-5	0776-27-1686
オーディオ片山	福井市中央1-5-3	0776-23-3360
加賀マイコンセンター	加賀市大聖寺耳聞山町60	
マルツ電波	福井市豊島2-74	0776-21-2360
北陸バイトショップ	富山市五福5区3216 市五福	0764-33-5176
無線パーツ	富山市布瀬2区759-4	0764-21-6822
ウスヰパソコンセンター	富山市総曲輪3-6-3	0764-21-4181
マイコンプラザ藤井	富山市音羽町2-5-15	0764-91-3887
インパルス	富山市五番町4-10 西野ビル	0764-91-2212
無線パーツ高岡店	高岡市永楽町2-4	0766-25-5045
麻地時計店	黒部市三日市3118	0765-52-0103
シーエル商会	敦賀市松島2-8-39-8	07702-5-2799
システムイン福井	福井市豊島1-3-1	0776-20-3485
タケウチ電子㈲	豊橋市大橋通2-132-2	0532-52-2684
㈱清水パーツセンター	清水市江尻東2-1-3	0543-65-1644
河合無線㈱	津市丸之内31-20-2F	0592-26-0111
河合無線㈱	伊勢市一之木2-12-4	0596-24-8111
文化センター白揚	四日市市諏訪栄町3-7	0593-51-0711

近　畿

店名	住所	電話
和歌山コンピュータシステムズ	和歌山県伊都郡かつらぎ町妙寺464	07632-2-7000
ＡＶシステムサービス	南河内郡狭山町大野台5-1-10	0723-66-6571
システムプラザ東邦	奈良市三条町474 福森ビル4F	0742-26-0051
カツミデンキ	守山市梅田町223-3	07758-3-6161
㈱ジェプロ	京都市上京区烏丸今出川南西角第2高橋ビル5F	075-415-0885～6
ヒエン堂本店	京都市下京区寺町通綾小路南西角	075-361-0371
オーム社書店	京都市中京区河原町通四条上ル	075-221-0280
システムハウス洛北	京都市左京区上高野古川町14	075-722-7100
谷山無線電機3号店	京都市下京区寺町通高辻上ル	075-351-2102
坂口テレビサービス	大津市大萱1-2-29	0775-45-4262
パルス	守山市小島町935-130	0775-83-5585
ヒバリヤ書店 本社	東大阪市足代2-29	06-722-1121
ハード&ソフトNDK	大阪市西区西本町1-5-3 扶桑ビル	06-543-2288
日本マイコン学院	大阪市北区中崎西1-4-22	06-374-0848
阪急百貨店 梅田店	大阪市北区角田町8-7(5F)	06-361-1381
旭屋書店 本店	大阪市北区曽根崎2-12-6	06-313-1191
〃 ナンバ店	大阪市南区難波新地6番町12 なんばCITY B2	06-644-2551
ナンバブックセンター	大阪市南区難波新地5-42	06-644-5501
紀伊国屋 梅田店	大阪市北区芝田阪急三番街	06-372-5821
共立電子産業	大阪市浪速区日本橋5-7-19	06-644-4446
高電社パソコンセンター	大阪市北区梅田1-11-4 駅前第4ビル6F	06-341-3371
コンピューターサービス	大阪市北区梅田3番地 駅前第3ビル	06-345-3351
東亜エレジャック	大阪市浪速区日本橋5-11-7	06-644-0111
上新電気J&P	大阪市浪速区日本橋5-6-7	06-644-1413
上新電機 日本橋1ばん館	大阪市浪速区日本橋5-1-11	06-644-1813
上新電機 日本橋5ばん館	大阪市浪速区日本橋4-12-4	06-644-1513
上新電機 阪急三番街店	大阪市北区芝田1-1-3	06-372-6912
MTK電子	尼崎市昭和通4-120	06-413-0188
マイコン・ジム	神戸市中央区三の宮1-3-4	078-392-4671
ジュンク堂書店	神戸市中央区三の宮1-6-18	078-392-1001
イナハラ事務機	神戸市中央区三の宮1-8-1-148	078-351-2145
マイコンの店 どらえもん	神戸市中央区三宮町1-8-1 サンプラザ10F	078-392-2004
システムイン神戸	神戸市中央区磯辺通2-1-13	078-232-0001
㈱コンピュートピア	神戸市東灘区岡本2-13-16	078-452-5600
スーパーブレイン神戸店	神戸市中央区三宮町1-8-1 サンプラザ8F	078-391-0181
COSMOS 明石マーベル	明石市西明石南町1-10-13	078-923-5536
星電パーツ 加古川店	加古川市加古川町笹原町2-4	0794-21-0551

中国・四国・九州

店名	住所	電話
エレパーツニシマル	宇部市上町1-2-25	0836-33-4422
ＯＡセンターヒロケン	下関市岬之町9-6	0832-24-0707
サンロードナンバーワン	下関市竹崎町4-2-30	0832-34-3660
電子システム	北九州市小倉北区魚町2-6-1	093-531-9581
倉敷ハムセンター	倉敷市美和1-4-23	0864-25-1300
MTK タネモリ	広島市中区西十日市町2-1	0822-46-8494
ダイイチ	広島市中区紙屋町2-1-18	0822-47-9111
旭電器ビデオパソコンセンター	三原市宗郷町115	08486-2-7169
京屋書店	宇部市松島町16-25	0836-31-2323
松本無線パーツ	広島市中区銀山町2-6	0822-43-4451
第！産業	広島市中区紙屋町2-1-18	082-247-5111
アマチュアTV どさんこ	三次市粟屋町2236	08246-2-1049
エレホビー サムラ	徳山市浦山開作8170	0834-32-0880
山菱電子販売	徳島市助任橋1-22 清水ビル2F	0886-23-7183
デジック興安	新居浜市新須賀町2-2-18	0897-34-8286
英弘チェーン 千舟店	松山市千舟町5-5-5	0899-41-8751
〃 表町店	岡山市表町2-7-12	0862-24-5886
〃 高知店	高知市旭町2-52	0888-23-2231
西日本マイコンセンター	高松市多賀町2-8-22	0878-33-8673
英弘チェーン 倉敷店	倉敷市笹沖下田471-1	0867-25-8645
〃 高松店	高松市丸亀町2-14	0878-51-7222
松山四国電業	松山市一番町4丁目	0899-43-0691
野田屋電機	高松市丸亀町1-3	0878-51-4545
電化センター	高松市天神前4-35	0878-62-6077
山菱電子販売㈱ 高松営業所	高松市中新町2-9 富士ビル2F	0878-61-7266
都電機商会	徳島市寺島本町東2丁目	0886-22-2134
コンピュータ・プラザ高松	高松市屋島西町1464	0878-43-1304
マイコンハウス コンド㈱	今治市喜田村3881	0898-47-1337

扱店一覧表

大坂屋・上智デルキ ン	大川郡大内町三木松出晴	08792-4-0234
高知マイコンセンター	高知市南御座字札場9-6	0888-84-3750
パソコンパーラーO&F 新電	川之江市川之江町357	0896-58-2138
デジック	松山市本町6-6-7 ロータリー本町ショップ	0899-41-6270
防府パソコンセンター	防府市中央町4-9	0835-23-8117
キーボード	防府市栄町1-9-2	0835-38-2880
北九無線パーツ部	北九州市小倉北区魚町2-2-17	093-551-7306
紀伊国屋書店 熊本店	熊本市下通1-7-18	0963-22-5531
金龍堂書店	熊本市上通町8-19	0963-54-5963
ブックスまるぶん	熊本市上通町5-1	0963-52-5665
春苑堂ブックプラザ	鹿児島市東千日町1-12 高島屋プラザ別館	0992-25-3200
松藤	熊本市下通り1-9-1	0963-54-9111
㈱IVC宮崎マイコンショップ	宮崎市宮脇町89-4	0985-27-4326

関 東・甲信越

アルゴジャパン	上伊那郡辰野町宮所19	02664-2-2022
第一無線工業	上田市天神2-4-60	0268-27-6624
データコントロール吉村	伊那市八幡町1980-1	02657-8-6448
マイコンセンター茅野	茅野市塚原1-3-23	02667-3-5270
マイコンプラザ上田	上田市天神1-9-3 進光ビル3F	0268-27-8778
マイコンショップ松本	松本市高宮西2-14	0263-27-1903
鈴木電興 長野支店	松本市野溝木工1-3-16	0263-26-8662
システムイン信州 長野OAセンター	長野市南石堂町1282	0262-27-6136
〃 松本OAセンター	松本市深志1-2-11 昭和ビル内	0263-36-5301
〃 佐久OAセンター	佐久市猿久保 ビューティー浅間	02676-8-2777
松本電子部品商会 長野店	長野市居町7	0262-44-2050
松本電子部品商会 飯田店	飯田市今宮町456	0265-24-6871
アサヒ電子部品	長野市栗田859	0262-26- 045
中央無線商会	伊那市中央区本町通2-6	02567-8-7628
マイコンショップ諏訪	茅野市塚原1-12-6	02 67-3-3248
マイコンセンター丸信	諏訪市末広6-12	02665-2-3287
エイム・システムズ	甲府市住吉5-6-31	0552-32-1451
サン・システムズ	甲府市中央2-9-5	0552-32-1391
神戸無線	須坂市大字須坂1230-50	
計測技研	宇都宮市桜2-1-30	0286-24-5010
オフィスオートメーション研究所	宇都宮市桜3-2-17	0286-24-5010
橋本デンキ販売㈱	宇都宮市馬場通り1-1-10	0286-22-0301
㈱クリマデンキ 本庄店	本庄市七件町2701-74	0495-24-3219
東芝パソコンサロン大宮	大宮市櫛引町2丁目285-1 立之ビル1F	0486-51-1100
㈱マツダ商事	桐生市稲荷町3-20	0277-47-1231
神戸楽器	群馬県甘楽郡下仁田町大字下仁田360-13	02748-2-3059
㈱ニューライフ	竜ヶ崎市出し山145	02976-4-2301
㈲大竹文具店	伊勢崎市本町1番18号	0270-25-5550
群栄産業	伊勢崎市日及出町76	0270-23-4818
群馬システム通信	前橋市三俣町2-17-10	0272-33-8626
㈱クリマデンキ・デンキセンター	前橋市本町1-2-21	0272-32-0110
スガヤ電機	前橋市天川町1667-22	0272-63-2559
坂庭エムセール㈱	高崎市飯塚町17-1 高義ビル5F	0273-63-3234
山三事務機㈱ 高崎営業所	高崎市問屋町2-8-11	0273-61-8848
イイノ技研㈱	前橋市西片貝町5-15-21	0272-24-8890
伊勢崎Byteショップ	伊勢崎市今井町755	0270-23-2302
伊藤商事㈱ 前橋店	前橋市問屋町2丁目11-1	0272-51-5255
㈱広井	伊勢崎市今泉町2-401	0270-25-1415
関東スペースコントロールズ㈲	桐生市稲荷町4-16	0277-43-0132
サカキ	高崎市問屋町1-10-26	0273-62-1550
アサヒ商会	高崎市問屋町2-8-2	0273-63-1111
両毛通信㈱	足利市西砂原後町1199	0284-41-8695
光栄マイコンシステム	足利市今福町215	0284-21-2108
光栄マイコンシステム 足利店	足利市伊勢町2-1-15	0284-44-1581
山三事務機㈱ 高崎営業所	高崎市問屋町2-8-11	0273-61-8848
クリマデンキ 太田店	太田市飯田町市民会館	0276-46-0394
細田電器学園店 マイコン部	新治郡桜村吾妻3丁目 筑波研究学園都市	0298-51-2667
スズショー	黒磯市高砂町4-17	02876-2-1536
SIS-IN 水戸	水戸市城南町3丁目16-21	0292-26-9198
伊勢崎Byteショップ	伊勢崎市今井町755	0270-23-2302
サカキ駒形バイパス伊勢崎店	伊勢崎市連取町464-3	0270-23-5151
伊藤商事㈱ 前橋店	前橋市問屋町2丁目11-1	0272-51-5255
㈲モリタ無線	佐野市万町2891	0283-3-1611
民生電気㈱	高崎市問屋町2-4-5	0273-61-4540
㈱ヌマニウ	真岡市並木町3-2-5	02858-2-3375
システムイン宇都宮	宇都宮市本町13-17 塚田ビル4F	0286-21-1162
㈲アヅマ家電	佐野市植木町1166	0283-3-1224
紀伊国屋 新潟店	新潟市万代町1-3-30	0252-41-5281
㈱雄電社	長岡市表町2-3-3	0258-32-2646
㈱エスエフシー新潟	新潟市関屋田町1丁目13	0252-66-2233
㈱エスエフシー新潟長岡営業所	長岡市坂之上町2丁目5-14	0258-35-0189
㈲昭和商会	佐渡郡佐和田町東大通り	0252-7-3121
システムエース	三条市北新保2-1-40	02563-5-5431
コンピュータシステム	三条市荒町1-12-30	02563-5-5795
二葉デンキ商会	糸魚川市大字寺町331-1	02555-2-0514
ニーズ	北蒲原郡加治川村下坂町436	02543-3-2234
長岡ハムセンター	長岡市柏町1-5-17 松電ボウル1F	0258-32-8661
コスモス新潟	新潟市花園町1-3-7 花園ビル2F	0252-44-6328
システムイン新潟	新潟市東中通2番町291 司ビル2F	0252-25-0895
柏崎マイコンスクール㈱科学技術教育協会	柏崎市西本町1-6-5	02572-4-7895
㈱NSI 本町店	上越市本町5丁目3-32	0255-22-0283
㈱NSI 木田店	上越市木田1310 木田中ビル1F	0255-22-2441
㈲小松電機	新潟県南魚沼郡六日町76-19	02577-2-2252
新潟ハムセンター	新潟市堀の内45	0252-45-4939
新潟マイクロコンピューター	新潟市鐙1丁目1-7 田辺ビル2F	0252-43-0818
BSNOA プラザ	新潟市東大通1丁目3-1 帝石ビル別館	0252-43-0334
川又書店	水戸市泉町2-2-31	0292-24-2047
マイコンショップ(ユニカ)	水戸市城南3-3-26	0292-26-1927
たけのうら電器	沼田市東原新町1820	0278-2-3653
デンケン	水戸市根本1-66-1	0292-21-9674
マイコンショップCAT	日立市鹿島町1-14-7 銀座通り永中ビル2F	0294-24-2777
㈱イーエスデイラボラトリ 筑波事業所	筑波郡谷田部町小野崎南小池180-1	0298-51-8070
マイコンショップ ベース	水戸市南町1-2-22 みとサントピア6F	0292-26-3867
㈱システムポート筑波	新治郡桜村天久保1-18-3	0298-52-4141
㈲北島電気商会	下館市女方5番地	02962-8-0050
井沢電器	佐野市伊賀町635-2	0283-3-2690
スズショー	黒磯市高砂町4-17	02876-2-1536
㈲中島無線電機	足利市大町442	0284-41-3327
十字屋電子システムセンター	松本市中央3-2-7	0263-35-3471
マイコンHAT	東茨城郡美野里町羽鳥2576	02994-6-3511
ベルコム	日立市千石町1-3	0294-37-1331
システムポート筑波	新治郡桜村天久保1-18-3	0298-52-4141
なみき書店	鹿島郡神栖町知手すずらん通り	02999-6-1855
マルスズ電機㈱	取手市新町5-17-17	02977-4-1311
ムラウチ	八王子市大和田町5-1-21	0246-42-6211
久美堂本店	町田市原町田6-11-10	
久美堂 小田急店	町田市原町田6-12-20 小田急8F	0427-23-7088
シーガル	八王子市中町7-7 西川ビル3F	0426-25-9960
コンピュータランド立川	立川市曙町1-19-3	0425-27-7037
システムインNITSUKO 立川	立川市錦町1-1-23	0425-27-3211
LAOX 吉祥寺店	武蔵野市吉祥寺本町1-20-3	0422-21-3437
LAOX 三鷹店	三鷹市野崎198	0422-32-3741
京葉システム㈱	千葉市新千葉2-1-5 小川第一ビル	0472-46-2196
タニカツ電機販売㈱	流山市江戸川台東2-7	0471-54-1111
関口商事	千葉県柏市旭町1-3-3	0471-44-3355
㈱アサノ	夷隅郡大多喜町大多喜駅前通り	04708-2-2647
アシーネ市川	市川市市川1-4-10	0473-26-3311
LAOX 船橋ららぽーと店	船橋市浜町2-1-1 ららぽーと2F	0474-34-3971
東京旭屋書店 船橋店	船橋市本町7-1-1 船橋東武VIV5F	0474-24-7331
LAOX 市原店	市原市五井町2343-1	0436-21-5331
家電のササキ	鴨川市横渚292	04709-2-1323
ナショナルの田島	行田市行田3-3	0485-56-2623
クリマデンキ 本庄店	本庄市七件町2701-74	0495-24-3219
パナック	港区西新橋2-3-2 ニュー栄和ビル	03-595-1057
パソコンロビーNTK品川	港区高輪3-25-29 前川ビル1F	03-445-5071
富久無線	千代田区外神田1-13-1 2F	03-255-1721
三省堂本店	千代田区神田神保町1-1	03-293-3441
旭屋書店 渋谷店	渋谷区宇田川町	03-476-3971
コンピュータブックセンター	渋谷区渋谷2-22-10 タキザワビル8F	03-499-3064
大盛堂書店	渋谷区神南1-22-4	03-463-0511
紀伊国屋 新宿店	新宿区新宿3-17-7	03-354-0131
紀伊国屋書店アドホック店2F	新宿区新宿3-17-7	03-354-0131
書泉グランデ	千代田区神田神保町1-3	03-295-0011
COM	千代田区神田佐久間町1-8-4 ニュー千代田ビル	03-251-8951
マイクロコンピュータSIHNKO	千代田区神田佐久間町1-8-4 ニュー千代田ビル	03-251-1523
ワールドゼア	新宿区西新宿1-16-10	03-342-8523
LAOX 新宿店	新宿区新宿3-15-16	03-350-1241
LAOX 本店	千代田区外神田1-2-9	03-253-7111
真光無線㈱	千代田区外神田1-15-16 秋葉原ラジオ会館7F	03-253-5085
ロケット本店	千代田区神田佐久間町1-14-1	03-257-0606
ロケット 3号店	千代田区外神田1-4-6	03-257-0347
水谷電機工業㈱	千代田区外神田1-15-6	03-253-4341
普賢電子	中野区中野3-34-24 千野ビル3F	03-382-8800
小沼電気	千代田区外神田1-15-16	03-251-2311
KOYOバイトショップ	千代田区外神田1-15-16 ラジオ会館	03-255-6504
関東Byteショップ	千代田区外神田1-15-16 ラジオ会館4F	03-253-5264
角田無線電気パーツガーデン	千代田区外神田3-13-8 3F	03-253-8121
ツクバマイコンセンター	千代田区外神田1-8-1	03-255-2741
亜土電子	千代田区外神田3-10-7 第2北沢ビル	03-255-9515
第一家庭電器マイコン相談室	千代田区外神田1-15-16 秋葉原ラジオ会館6F	03-253-7948
〃 マイコン相談室	新宿区西新宿1-26-2 新宿野村ビルB2F	03-346-2381
旭屋書店 池袋店	豊島区西池袋1-1-25 東武デパート10F	03-986-0311
池袋西武マイコンコーナー	豊島区南池袋1-28-1 西武百貨店内	03-981-0111
ヤマギワ テクニカ店	千代田区外神田4-1-1	03-253-2111
〃 世田谷店	世田谷区千歳台3-20-1	03-483-3111
丸善㈱	中央区日本橋2-3-10	03-272-7211
八重洲ブックセンター	中央区八重洲2-5-1	03-281-1811
東武デパートマイコンコーナー	豊島区西池袋1-1-25 東武デパート7F	
竜文堂書店 駅ビル店	大田区山王2-1-5	03-775-3851
日本インターナショナル整流器	品川区西五反田7-22-17 TOC8F	03-494-2411
一清電気	品川区戸越2-5-5	03-783-9301
東急ハンズ	渋谷区宇田川町12-18 東急ハンズ内6F	03-464-4604
中本無線	神奈川県津久井郡城山町川尻1007	0427-82-5774
㈱こくぼ	横浜市緑区中山町632-2	045-934-5115
ベーシックシステム	横浜市西区南幸2-7-12 石辺ビル5F	045-314-4649
㈱L商会 西口鶴屋町店	横浜市神奈川区鶴屋町1-1	045-312-2055
光栄マイコンシステム日吉店	横浜市港北区日吉本町1876	044-61-6861
トヨムラ 横浜店	横浜市中区松影町1-3-7	045-641-7741
LAOX 厚木店オーディオ館	厚木市中町4-16-6	0462-22-2722
足柄ハムセンター	神奈川県足柄上郡大井町金子1624-3	
緑ハムセンター	横浜市緑区つつじヶ丘36-10	045-983-8216
ヤマギワ 横浜店	横浜市中区羽衣町2-5	045-261-2111
工人舎	横浜市中区松影町2-7-21	045-662-0688
L商会 横浜西口店	横浜市神奈川区鶴屋町1-1	045-312-2055
L商会 上大岡店	横浜市港南区上大岡西1-10	045-844-2055
L商会 藤沢店	藤沢市藤沢572	0466-26-7655
L商会 南部市場店	横浜市金沢区富岡2555	045-771-1911
L商会 小田急相模原店	相模原市相南4-24-32	0427-48-3399
㈱太田デンキ	新座市東北2-31-15	0484-73-1511

月刊マイコン オリジナル・ソフト

このプログラムソフトは月刊マイコンで記事、特集として掲載され好評を博したものを、読者サービスとしてカセット・サービスしたものです。他にないユニークなプログラムが揃っています。

テープ名	内容	定価	機種名	言語	注文書
リアルタイム SUPER STAR TREK 1735	今までのTREKゲームの常識をうち破った傑作。ワープ航法，長距離レーダー始動防御スクリーン作動，積載コンピュータをフル活用して，クリゴンと頭脳戦争だ！	3,000円	PC-8001	B	
みみずの滝のぼり 1736	迫りくるゲジゲジの大群に果敢に立ち向かうミミズの勇士。でもゲジゲジにつかまると，ゲジゲジが次々と成長し状況悪化。ゆけミミズ戦士よボーナスの日まで！	3,000円	PC-8001	B	
コードネーム自動表示 1737	ピアノ・ギター楽譜のコード進行チャート，コード修正をスピーディに！ピアノとギターが同時に表示され，またコードを楽譜化して見ることが出来ます。	3,000円	PC-8001	B	
インデアン・ポーカー 1738	PCとあなたのしのぎを削る賭け金の競い合い。強気になったり，弱気になったり，いかにも人間らしくふるまうPC。あなたとPC，どちらが破産？	3,000円	PC-8001	M B	
SUPER卓球ゲーム 1739	本物そっくりの卓球ゲーム，ラケットスイングができ，打球角度を自由にコントロールできます。コンピュータ相手にパーフェクト試合ができたらあなたは天才！	3,000円	PC-8001	M B	
エイリアンビリヤード 1740	エイリアン相手にビリヤード！あなたのたくみなスティックさばきで，見事にエイリアンを撃退してください。マシン語& BASIC オートスタートです。	3,000円	PC-8001	M B	
少年とエイリアン 1742	宇宙元年8001年，月面基地に生き残った少年3人と異星人との激しい戦い。勝ち残った少年だけが，栄光のエイリアンレースに参加できます。	3,000円	PC-8001	M B	
成績処理 1743	①集計表（合計，平均，標準偏差），②ヒストグラム各種，③素点表（順位，偏差値を含む），④偏差値表（各教科の偏差値とその散らばり），⑤個人向けカード，⑥順位表以上の処理が出来ます。	3,000円	PC-8001	B	
ピラミッドとミイラ 1790	オセロとルービックキューブを組合せた様なゲームで，系統的に考えていかないとなかなか完成しません。たとえ完成出来ずGIVE UPしてもあとはPC−8001が考えて完成させてくれます。	3,000円	PC-8001	M B	
ALIEN LAND 1957	人類の平和を守るため，ロボットをうまく操縦して下さい。エイリアンを避けて，エネルギー鉱石を一つでも多く取って下さい。アタックエイリアンに要注意。	3,000円	PC-8001	B M	
スーパー・ムービング・ブロック 1958	ワーピング・ラケット，攻撃するエイリアン，天じゃステーションが笑っている。ワザ有り，運有り，度胸有り！オールマシン語でスピードも抜群。	3,000円	PC-8001	B M	
ウッドペッカー 1959	緑の木立ちにウッドペッカーやって来て次々と木を倒してしまいます。さあ，あなたはどれだけウッドペッカーを生け捕りにできるか？	3,000円	PC-8001	B	
二次方程式解法テクニック 1961	四則計算から正負の計算、文字式、一次二次方程式にいたるまでの解法を、計算の仕方と基本を重視して展開表示します。式を設定するのはあなたです。	3,000円	PC-8001	B	
オイチョ・カブゲーム 1744	日本古来のカードゲーム。胴元はもちろんマイコンです。よく考えて勝負して下さい。熱くなって身ぐるみはがされないように。	3,000円	MZ-80B	B	
財務会計(SP-6010) 1732	帳簿の整理をお手伝いします。簿記3級程度で小規模商店向きです。A面に会計処理とB面に決算処理の二つにわけてありメンテナンスが容易で操作ミスも防げます。現金収納帳，経費明細帳，残高試算表などの発行が出来ます。	6,000円	MZ-80K/C	ディスク B	
家計簿 1733	奥様の帳簿作成の効率アップ!!誰でも簡単に使える実用プログラム。月に7650件のデータ入力が可能。毎日カセットテープへデータ保存。月末にプリントアウトでOK。	3,000円	MZ-80K/C	B	
販売管理V2.0 (SP-6010) 1734	売上帳，仕入帳，仕入先元帳，得意先元帳，請求書の発行が出来ます。データが発生する度にディスクへ登録しますので電源事故を未然に防げる実用プログラム。	3,000円	MZ-80K/C	ディスク B	
在庫管理(SP-6110) 1741	扱い易い実用プログラム。指定項目のみの在庫表，入出庫明細表，出庫計算書などがプリントアウト可。またディスケットを増やすことで商品管理数は無限です。	4,000円	MZ-80K/C	ディスク B	
THE ケンドウ 1458	あなたとマイコンとの剣道試合。上段のかまえ，下段のかまえでマイコン剣士をやっつけろ。ハイスピード・ロースピード調整OK。BASIC & マシン語。	3,000円	ベーシックマスターL2	B M	
バッティングゲーム 1459	ストレート・カーブ・フォーク・パーム等，実際にバッターボックスに立った感じが味わえます。プロ級，アマチュア級，スピードボール練習用，カーブ練習用の4段階あります。BASIC & マシン語。	3,000円	ベーシックマスターL2	B M	
パルルルーン ポンパー 1461	降下してくる風船を撃ち落として下さい。きちんと当たらないと爆弾が落ちてきて地面に穴をあけてしまいます。また飛んでいる飛行機に弾が当たるとミステリーポイントがもらえます。	3,000円	ベーシックマスターL2	B	
LⅡモニタ解読用逆アセンブラ 1460	ベーシックマスタのモニタ研究に御利用ください。実用上，十分な速さで逆アセンブラしてくれます。	3,000円	ベーシックマスターL2	B	
モニタ解読用逆アセンブラ 1453	便利な各種コマンド付き。L3ユーザーには必需品でしょう。	3,000円	ベーシックマスターL3	B	
THE ギャング 1961	大金が眠る豪邸の金庫へたどりつくには、数々の迷路のワープトンネルを利用して突破しなければならない。超一流のギャングである君の行く手を待つのは、大金持ちへの道か、冷たい牢獄か、はたまた大爆発か！	3,000円	PA-7010 パソピア	T B	

※PC-8001用ソフトはPC-8801のN-BASICモードでも使用できます。
※ベーシックマスターL2用ソフトはベーシックマスタJr.にも使用できます。

Dempaマイコンソフトテープ

テープ名	内容	定価	機種名	言語	注文書
(グラフィック) スキー・ゲーム 1788	迫り来る木立ちをよけながら、直滑降、回転！ 10のスキーコースをすべて滑りこなせれば、あなたは名スキーヤー。	3,600円	MZ－80B	B	
(グラフィック) 海賊ゲーム 1789	「艦長、海賊船にねらわれています。」逃げまどいながら、海賊船を3回撃破しないと乗りうつられますゾ！	3,800円	MZ－80B	B	
(グラフィック) 鮫うちゲーム 1791	あなたは、スキン・ダイバー。しかももっとも危険なサメ打ちのプロハンター。サメをやっつけるか、それとも食われるか。潜水技術とモリ打ちのウデで海の平和を守れ！	3,800円	MZ－80B	B	
ビンゴゲーム 1792	コンピューターを相手に交互にマス目をつぶしていって、先につぶした数字の列が5列できた方が勝ち。	3,800円	MZ－80B	B	
オカルトハウス 1080	宝の埋まっている屋敷を発見。怪人に捕ったり部屋に閉じ込められない様に注意して財宝を拾いあげて下さい。㊄G－RAMNo.1	3,600円	MZ－80B	B	
プレイボーイゲーム 1081	箱入り娘を外へつれだすゲームです。時間オーバーすると泣いてしまいます。㊄G－RAMNo.1	3,800円	MZ－80B	B	
水戸黄門ゲーム 1082	攻めてくる悪漢どもを助さん、角さんを動かして、やっつけてください。無事に旅を続けられるのもあなたの腕しだい。 ㊄G－RAMNo.1	3,600円	MZ－80B	B	
神経衰弱 1084	6人までの人数で遊べます。1人の時は80Bがお相手！さあ、あなたの記憶力はだいじょうぶかな。 ㊄G－RAMNo.1	3,400円	MZ－80B	B	
タコタコあがるな 1085	穴から飛び出たタコを網で捕えるゲーム。油断すると、すぐ逃げられてしまいますよ。㊄G－RAMNo.1	3,600円	MZ－80B	B	
スーパーブロックくずし 1087	おなじみTVゲームの元祖。DOWN BLOCK、MOVING BLOCK、CRAZY BLOCKの3種類が楽しめます。	3,400円	MZ－80B	M	
駆逐艦撃沈ゲーム 1088	潜水艦から魚雷を発射して駆逐艦を撃沈して下さい。ただし魚雷は20発しかないので正確に。 ㊄G－RAMNo.1	3,600円	MZ－80B	B	

ゲーム・実務トレーニングソフトシリーズ for Micro-8

テープ名	内容	定価	機種名	言語	注文書
ダービー 1900	単勝馬券で5名まで参加OK！賭金と対象馬が決まると予想配当金が表示され、出走！ディスプレイ左側から馬が走る。ゴールすると当り馬券と配当金が出る。	3,000円	MICRO－8	B	
オセロ 1901	マイコンを相手にオセロゲームはいかが？相手はけっして熱くならず、クールに駒をすすめてくる。マイコンの頭脳をあなたの能力はこえられるか！	3,000円	MICRO－8	B	
月面着陸 1902	あなたは月面着陸船の操縦士。引力、エンジン噴射、エネルギー量など、安全着陸させるには高度なテクニックと計算が必要だ。成功をいのる。	3,000円	MICRO－8	B	
アルデバラン #1 1903	スタートレックをしのぐBASICゲームの決定版。月刊マイコン'79年1月号で発表され、大人気を集めたSFストーリー・ゲームのFM8版。	3,600円	MICRO－8	B	
スタートレック 1904	マイコンゲームの古典的名作、スタートレックのMICRO－8版。ニックキ敵"クリゴン"を光子魚雷と波動砲を使って撃破せよ！すすめ！エンタープライズ！	3,600円	MICRO－8	B	
アニマルレッスン 1905	マイコンは動物の知識を増やそうと必死。「飛びますか？」「走りますか？」etcと聞いてくる。あなたが教え込んでマイコンを動物学者にしてしまおう。	3,000円	MICRO－8	B	
頭の体操 No.1 1906	計算能力、判断力、敏速性、記憶力の四つのジャンルをテストし、総合得点が表示される。仲間と集まって楽しむには最高のゲーム。熱くなることうけあい！	3,200円	MICRO－8	B	
ニュートン法 1907	方程式f(X)＝0の近似値解を求めるために、微分を使って算出するソフトウェア方程式を入力すれば、たちどころに解をアウトプットしてくれる。	3,000円	MICRO－8	B	
多角形の面積計算 1908	測量用の実用ソフトウェア。もっともポピュラーな多角点測量のデータを計算しデータを求めるプログラム。データ修正もディスプレイ上で行なうことができる。	3,000円	MICRO－8	B	
多元連立方程式 1909	二元以上、27元までの一次方程式をディスプレイ上に数値を置くだけで短時間に解くマイコンならではのソフトウェア。解法は消去法で行なわれる。	3,000円	MICRO－8	B	
表集計 1910	表作成に必要な数値を入力することにより、XYの表計を計算してくれるソフトウェア。データはカセットに記録でき、集計は画面、プリンタどちらでも可能。	3,600円	MICRO－8	B	
SS計算 1911	学校などで使われる成績処理プログラム平均点、順位、偏差値などの統計をマイコンが処理してくれる。プリンタに打出すこともOK。	3,000円	MICRO－8	B	
英会話レッスン 1912	マイコンがランダムに出題してくる設問にキーボードで解答。知らず知らずのうちに実力がつく実用英会話レッスンのソフト。点数による実力判定も可能。	3,000円	MICRO－8	B	
価値判断 1913	マイコンなら入力されたデータにより色付けなしの価値判断が可能。物品購入、人材選び、競馬予想など、活用は無限。10品種、10項目のデータ入力ができる。	3,400円	MICRO－8	B	

実用トレーニング・ソフト編 for PC-8001

テープ名	内容	定価	機種名	言語	注文書
(32K) 機械語データーベース 1745	コンピュータをデータ・バンクとして利用するためのプログラム。コマンドを使い分けることにより、ワード・プロセッサとしても活用することが可能。(マニュアル付)。	6,000円	PC-8001	M	
(32K)財務管理 1746	日常業務で発生する勘定科目の貸借に応じた金額を仕訳してカセット・テープにデータをたくわえ、月末に試算表を計算し、プリンタに打出すプログラム。(マニュアル付)	8,000円	PC-8001	B	
(32K)在庫管理 1747	総合的な倉庫管理を行なうためのプログラム。日常の商品の入出庫を管理し、発注の指示もだす。在庫総額、仕入総額、売上総額のプリント・アウトも可能。(マニュアル付)	4,000円	PC-8001	B	
給与計算 1748	社員数50名までの給料計算ができる。さらに支給に必要な紙幣、硬貨の枚数を計算し、社員用及び会社用の明細書の作成も行なうプログラム。(マニュアル付)	4,000円	PC-8001	B	

Dempaマイコンソフトテープ

テープ名	内　容	定価	機種名	言語	注文書
(32K)顧客管理 1749	顧客数250,項目数は1～8まで可能。自店の使用目的に合わせて使用できる汎用ソフト。プリンターによる宛名印刷も可能。家庭の住所録としても便利。(マニュアル付)	4,000円	PC-8001	B	
(32K)仕入管理 1750	毎日の仕入データを商品別、仕入先別に入力すると、月間の累計を計算し、プリントアウトする。商品は最大100,仕入先は20社までOK。カラー表示。(マニュアル付)	3,500円	PC-8001	B	
(32K)販売管理 1751	100品目の商品、20件の得意先を対象として、商品の販売を管理する。当月売上げを商品別、顧客別に分類し、プリンタに出力することも可能。(マニュアル付)	3,500円	PC-8001	B	
(32K)請求・納品書作成 1752	**販売管理プログラムと組合わせて使うプログラム**。「販売管理」で作成したデータをもとに、客先別の納品書及び請求書を一般的フォーマットでプリント。(マニュアル付)	3,500円	PC-8001	B	
(32K)見積書作成 1753	あらかじめ登録された200品目以内の商品の中から、任意に100品目指定選びだし見積書を作成する。品目指定のみで単価や合計はマイコンにおまかせ。(マニュアル付)	4,000円	PC-8001	B	
連立方程式計算 1754	26元までの複雑な方程式をたちどころに計算してくれるプログラム。計算結果はプリンタに打出しできる。	3,000円	PC-8001	B	
KEY INPUT トレーニング 1755	パソコン時代を迎え、スピーディーで正確なキー・インプットは、どうしても必要なテクニック。ノーマル・カナ・ソフトの各モードで練習ができるプログラム。	3,000円	PC-8001	B	

新作実務トレーニングソフトシリーズ

テープ名	内　容	定価	機種名	言語	注文書
売掛金管理 (32K) 1944	このプログラムは、100件までの客先の〆日、支払日、売上額、入金額を登録しておき入金状況を管理することを目的としています。(マニュアル付)	3,500円	PC-8001	B	
買掛金管理 (32K) 1945	このプログラムは、100件までの仕入先の〆日、支払日、仕入金額、支払金額を登録し支払状況を管理することを目的としています。(マニュアル付)	3,500円	PC-8001	B	
金種計算 1946	最大500人、1人、三億円までの金額についてそれぞれの金種を計算して、CRT画面又はプリンタに出力できます。(32K実装時)	3,500円	PC-8001	B	
クレジット計算 1947	クレジット計算をパソコンで計算させてみませんか。各回数に応じた手数料の率を登録しておいて商品データと回数、ボーナスを入力し毎月の支払額を計算する。	3,000円	PC-8001	B	
手形管理 (32K) 1948	100件の得意先に対して、手形の起算日、支払日、額面金額を登録しておき、支払日までの残り日数、支払期日を管理し、不渡りを防止することを目的としています。(マニュアル付)	3,500円	PC-8001	B	
受注管理 (32K) 1949	100件の得意先それぞれの注文明細を登録しておき、納品済の品物、未納の品物に分類して、納入日、納入予定日、所要納期等をCRT画面又はプリンターに出力できます。(マニュアル付)	3,500円	PC-8001	B	
発注管理 (32K) 1950	100件の仕入先それぞれの発注明細を登録しておき、受領済みの商品と、そうでない商に分類して、納入日、納入予定日、納期のかかりすぎるものをCRT画面又はプリンタに出力します。(マニュアル付)	3,500円	PC-8001	B	
クロス集計 (32K) 1951	横最大10項目、縦20行までの表を5ページまで登録ができ、表の縦計、横計の集計を行う簡単なタテ、ヨコ計算で出来る多種多様の用途に応用が可能です。(マニュアル付)	3,500円	PC-8001	B	
カナ・ローマ字変換プログラム 1952	カタカナで入力した文章を、ローマ字に変更してCRT画面又はプリンタに出力する。(マニュアル付)	2,800円	PC-8001	B	
ローマ字・カナ変換プログラム 1953	ローマ字で入力した文章を、カタカナに変換してCRT画面又はプリンタに出力する。(マニュアル付)	2,800円	PC-8001	B	
車輛管理 (32K) 1954	50台までの車輛をそれぞれの走行距離、目的地、燃料補給回数、補給量等を登録して、車輛の管理をする企業の総務課向きのプログラムです。プリンタに上記データを出力することもできます。	3,500円	PC-8001	B	
工数計算 (32K) 1955	1ヶ月ごと、あるいは一週間ごとの各作業員の労働時間、作業内容を入力して各作業の工数を計算する。	3,500円	PC-8001	B	
原価計算 (32K) 1956	登録してある部品表の中から、必要な品物のみを引出して一つの作業の製造原価を計算する。	3,500円	PC-8001	B	

ゲーム・ソフト編 for PC-8001

テープ名	内　容	定価	機種名	言語	注文書
レーダーサーチ (32K) 1914	謎の宇宙人の手により地球上の主要都市が全て破壊されてしまった。最後のとりで、オアシス島にたてこもり、わずかに残されたレーダーを使って敵の侵略を防げるか。	3,000円	PC-8001	B M	
ばぐごん (32K) 1915	ピグミン君がちょっと目を放したすきにばぐごんにつかまってしまった。凶悪なゲドラ、ばぐこんに見つからない様に、ピグミン君を誘導して助け出してあげてくれ。	3,000円	PC-8001	B M	
シリウスF (32K) 1916	星間輸送船STA-1は、地球へ帰還する途中、時元断層に落ち込んで、未知の空間へ放り出されてしまった。異時元センサーを使い君は地球へ帰ることが出来るか。	3,000円	PC-8001	B M	
エアーライフル (32K) 1917	標的は3つ、距離はそれぞれ10m、20m、30mにセットされている。11発弾丸で君は何点かせげるかな？	3,000円	PC-8001	B M	
キングタイガーIII PART1 (32K) 1918	形勢逆転をもくろむドイツ軍総司令部は、キングタイガー戦車隊長の君に、とある鉄橋のそばで連合軍絶滅を命じた。さて勝利の女神はどちらに微笑むのか。	3,000円	PC-8001	B M	
バトルバルカン (32K) 1919	君は宇宙パトロールの隊員だ。今日も地球を守るためイプシロン星の侵入を防ぎに行かねばならない。気を付けろ、敵はするがしこいぞ。	3,000円	PC-8001	B M	
ビッグアステロイド (32K) 1920	アステロイドベルトへ迷い込んでしまった君！ わずかに残された補助ブースターを使い、アステロイドベルトを抜け出し、無事に地球へ帰り着くことが出来るか。	3,000円	PC-8001	B M	
ブラックホール (32K) 1921	ブラックホールに吸い込まれ、未知の世界に放り出されてしまった君！ 新兵器プロトン砲を使い、様々な困難を乗り越え、ホワイトホールへ脱出しろ。	3,000円	PC-8001	B M	
戦艦大和 (32K) 1922	第二次大戦も終りに近づき、特命を受けた戦艦大和は、沖縄に向け出発する。行く手をはばむ連合軍の攻撃をさけながら、君は無事にたどり着く事ができるか。	3,000円	PC-8001	B M	

Dempaマイコンソフトテープ

テープ名	内容	定価	機種名	言語	注文書
ドキドキすいか割り 1923	海水浴にやって来た君は、すいか割りを楽しんでいる。ところが、実はそのすいかの1つには、爆弾がかくされている。君はB国情報部のたくらみにひっかからず、うまく切り抜けられるか。	3,000円	PC-8001	B	
アスロック (32K) 1924	2101年、地球にはいたるところに海上都市が立ち並ぶ。超近代惑星になっていた。ところがある日突然謎の海底人の手によって各国の都市が破壊されてしまった。立ち上がれアクアマリン号。	3,000円	PC-8001	B M	
ワイルドスワット (32K) 1925	爆音を轟かせながら、町の中を我が物顔に駆け抜けてゆく暴走族。君は装甲パトロールカーMAD-Xに乗り込み、暴走族絶滅の為、今日もパトロールに出なければならない。	3,000円	PC-8001	B M	
プラネットバルカン (32K) 1926	宇宙歴2999年、いがみ合いの続くα星系とβ星系がついに第一次惑星間戦争に突入してしまった。相手を殺らなければ自分が殺られる。君はこの苛酷な試練に耐え、生き残ることができるか。	3,000円	PC-8001	B M	
ギャラクティカ 1 (32K) 1927	惑星探査船ギャラクティカは、惑星を調査中、突如、謎の宇宙人ゴモラの攻撃を受けた。ゴモラ軍との壮絶な闘いが始まる。スクリューカノン砲で敵を破壊しろ!	3,000円	PC-8001	B M	
ギャラクティカ 2 (32K) 1928	ゴモラ軍との戦闘で勝利を収めたギャラクティカの次の任務は、惑星イユリスの反乱鎮圧である。反乱軍は侵入者に対して無差別攻撃を加えてくる。急げ!	3,000円	PC-8001	B M	
ギャラクティカ 3 (32K) 1929	惑星デュオベータは太陽の重力変化の為絶滅の危機に瀕していた。ギャラクティカは惑星の内陸部にもぐり込み、デュオベータ人を救わねばならない。	3,000円	PC-8001	B M	
バトルファイアー (32K) 1930	せまりくる敵大船団。ターゲットスコープオープン! 自動追尾装置セットオン! 目標をまちがえるな。サブロックの数には限りがある。効率よく攻撃して、敵船団を壊滅させろ。	3,000円	PC-8001	B M	
CRTチェイサー PART 1 (32K) 1931	宇宙警備艇CCR1号は、領内警備のため、今日も10ヶのチェックポイントを周らねばなりません。しかしあちらこちらで機雷やミサイルがまちかまえています。上手に切抜けて高得点をかせいで戻ってきて下さい。	3,000円	PC-8001	B M	
CRTチェイサー PART 2 (32K) 1932	宇宙警備艇CCR1号は、外宇宙に配置転換になってしまいました。周りには宇宙機雷と目に見えない境界線があなたを待ち受けています。注意しながらチェックポイントを回って下さい。	3,000円	PC-8001	B M	
CRTチェイサー PART 3 (32K) 1933	PART1、2に飽きた方の為に、しばらくの間地球に戻ってみませんか? アッポ山には5000年も前の人類の埋蔵金が眠っています。低性能の探査装置をかついで、埋蔵金を探し当てて下さい。	3,000円	PC-8001	B M	
CRTチェイサー PART 4 (32K) 1934	埋蔵金探しの途中で、あなたは地底の迷路に迷い込んでしまいました。考えている間にも、どんどん酸素が少なくなって行きます。急いで脱出して下さい。	3,000円	PC-8001	B M	
マリンどんべえだあPART 1 (32K) 1935	あなたは海洋廃品回収船どんべII世号のチーフパイロットです。廃品識別装置を持っていないあなたは目を皿の様にしてゴミだけをかき集めなければならない。	3,000円	PC-8001	B M	
マリンどんべえだあPART2 (32K) 1936	どんべえII世号は外洋掃除の任務につきました。ところが何と、周りは海底火山だらけです。噴火をさけながら基地に戻らなければなりません。	3,000円	PC-8001	B M	
スカイどんべえだあPART1 (32K) 1937	大気汚染調査隊スカイどんべえは、今日も調査に出かけます。アッ! 危い。その雲はオキシダントのかたまりだ!!	3,000円	PC-8001	B M	
スカイどんべえだあPART2 (32K) 1938	1地区を2機で回ることになったスカイどんべえ。うっかりするとニアミスを起してしまいそうになります。あなたは如何にしてこの事態から逃れられるか。	3,000円	PC-8001	B M	
スペースどんべえだあPART1 (32K) 1939	あなたは監視船どんべえ1号の船長です。今日もアステロイドベルトのパトロールに出るのですが、エイリアンはするがしこくて惑星の陰にかくれています。目を皿の様にしてエイリアンを探して下さい。	3,000円	PC-8001	B M	
スペースどんべえだあPART2 (32K) 1940	エイリアンはどんべ1号を攻撃して来ます。奴らのミサイルはなかなか強力です。すきを見てお返しのミサイルで追いはらって下さい。	3,000円	PC-8001	B M	
タマツキゲーム 1941	キューの具合はいいか? クッションは全て計算したか? 角度と強さが決ったら、さあGO!! ハイテクニックを必要とするビリヤード・シミュレーションゲーム。	3,000円	PC-8001	B	
スペースランディング (32K) 1942	任務を終えて基地に戻って来た君は、これから宇宙ステーションマザーIに着艦しなければならない。突然降って来るいん石をかわしながら無事着艦することが出来るか。	3,000円	PC-8001	B M	
フューチャー (32K) 1943	宇宙連合の命を受けて、君はバード号に乗り込み、敵ガメロンを退治しなければなりません。突如出現するブラックホール、吸い込まれたら2度と生きては帰れない。君はガメロンを退治できるか。	3,000円	PC-8001	B M	

マイコン別冊

PC-8001/8801マシン語入門

塚越一雄 著

昭和57年7月20日発行

昭和58年1月10日第3刷発行

©1983 Printed in Japan

定価 1,300円(送料250円)

発行人 平山秀雄

発行所 (株)電波新聞社

郵便番号141 東京都品川区東五反田1-11-15

電話(03) 445-6111(大代) 振替東京5-51961

印刷所 大日本印刷(株)

製本所 (株)堅省堂